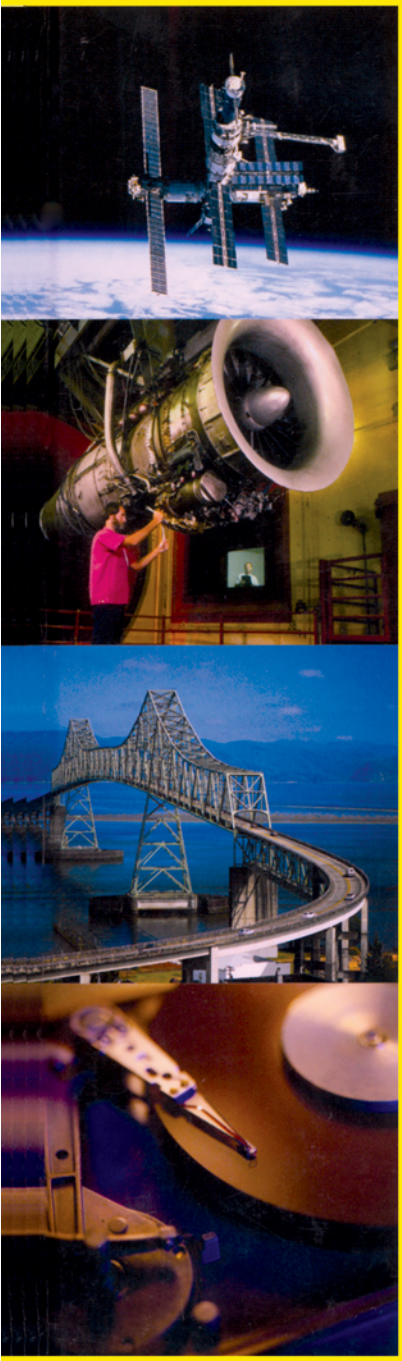
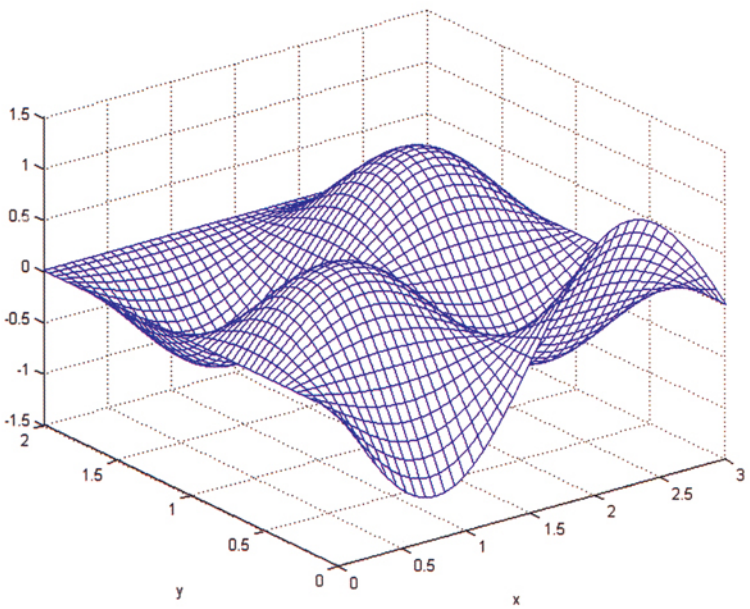
ELSEVIER
ACADEMIC
PRESS

STRESS, STRAIN, AND STRUCTURAL DYNAMICS



An Interactive Handbook of Formulas, Solutions, and MATLAB Toolboxes

BINGEN YANG



INCLUDES CD-ROM

Stress, Strain, and Structural Dynamics

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs and/or the documentation.

MATLAB is a registered trademark of The Mathworks, Inc.

Stress, Strain, and Structural Dynamics

An Interactive Handbook of Formulas, Solutions, and MATLAB Toolboxes

Bingen Yang
University of Southern California

ELSEVIER
ACADEMIC
PRESS

Amsterdam Boston Heidelberg London New York Oxford
Paris San Diego San Francisco Singapore Sydney Tokyo

Elsevier Academic Press
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
525 B Street, Suite 1900, San Diego, California 92101-4495, USA
84 Theobald's Road, London WC1X 8RR, UK

This book is printed on acid-free paper. ♾

Copyright © 2005, Elsevier Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone: (+44) 1865 843830, fax: (+44) 1865 853333, e-mail: permissions@elsevier.com.uk. You may also complete your request online via the Elsevier homepage (http://elsevier.com), by selecting "Customer Support" and then "Obtaining Permissions."

Library of Congress Cataloging-in-Publication Data
Yang, Bingen.
Stress, strain, and structural dynamics.
p. cm.
Includes bibliographical references.
1. Strains and stresses. 2. Structural dynamics. I. Title.
TA648.3.Y36 2004
624.1′7–dc22
2004022861

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN: 0-12-787767-3

For all information on all Elsevier Academic Press publications
visit our Web site at www.books.elsevier.com

Printed in the United States of America
05 06 07 08 09 10 9 8 7 6 5 4 3 2 1

Working together to grow
libraries in developing countries
www.elsevier.com | www.bookaid.org | www.sabre.org
ELSEVIER BOOK AID International Sabre Foundation

To my parents,
My wife Haiyan, and
My daughters Sonia and Tanya

This Page Intentionally Left Blank

Contents

Preface *xv*

Acknowledgements *xvii*

Chapter 1 Introduction 1
1.1 The Making of This Book 1
1.2 How to Use This Book 4

Part I Strength of Materials **7**

Chapter 2 Static Analysis of Euler-Bernoulli Beams 9
2.1 Getting Started 9
2.2 Beam Theory 11
2.2.1 Static Problem 11
2.2.2 Solution Methods 16
2.2.3 Fundamentals of Euler-Bernoulli Beams 23
2.3 Static Analysis by the Toolbox 25
2.3.1 System Setup 26
2.3.2 Response to External Forces 28
2.3.3 Response to Boundary Disturbances 33
2.3.4 Total Response 34
2.3.5 Exact Analytical Solutions 36
2.3.6 Other Useful Functions 40
2.4 Moments of Inertia of Beam Cross-Section Area 42
2.5 Quick Solution Guide 46
2.6 References 51

Chapter 3 Static Analysis of Bars, Shafts, and Strings 53
3.1 Getting Started 53
3.2 System Description 55
3.3 Static Analysis by the Toolbox 59
3.3.1 System Setup 59

3.3.2 Response to External Forces 60
3.3.3 Response to Boundary Disturbances 63
3.3.4 Total Response 65
3.4 Stepped Bars and Shafts 67
3.4.1 System Description 67
3.4.2 Solution by the Toolbox 68
3.5 The Distributed Transfer Function Method 74
3.6 Quick Solution Guide 78
3.7 References 83

Chapter 4 Buckling Analysis of Columns 85
4.1 Getting Started 85
4.2 Uniform Columns 87
4.2.1 Column Buckling Theory 87
4.2.2 Solution by the Toolbox 91
4.2.3 Eccentric Loading 96
4.2.4 Beam-Column Problem 100
4.2.5 Geometric Imperfection 104
4.3 Stepped and Nonuniform Columns 109
4.3.1 Constrained Stepped Columns 109
4.3.2 Nonuniform Columns 116
4.4 The Distributed Transfer Function Method 123
4.5 Quick Solution Guide 127
4.6 References 133

Chapter 5 Stress Analysis in Two-Dimensional Problems 135
5.1 Getting Started 135
5.2 Plane Stress and Strain 136
5.2.1 In-Plane Stresses 136
5.2.2 In-Plane Strains 142
5.2.3 Strain Rosette 147
5.2.4 Hooke's Law 149
5.2.5 Criteria of Failure 151
5.3 Quick Solution Guide 155
5.4 References 156

Part II Structural Mechanics 157

Chapter 6 Static Analysis of Constrained Multispan Beams 159
6.1 Getting Started 159
6.2 System Description 161
6.3 Solution by the Toolbox 166
6.3.1 System Setup 166
6.3.2 Response to External Forces 171
6.3.3 Response to Boundary Disturbances 176
6.3.4 Response to Support Settlement 178
6.3.5 Total Response 181
6.3.6 Influence Lines 183
6.4 The Distributed Transfer Function Method 189

6.5 Quick Solution Guide 194
6.6 References 200

Chapter 7 Static Analysis of Plane Trusses 201
7.1 Getting Started 201
7.2 System Description 203
7.3 Solution by the Toolbox 205
7.3.1 System Setup 205
7.3.2 Response to External Forces 210
7.3.3 Support Settlement 213
7.3.4 Fabrication Errors and Thermal Effects 216
7.3.5 Total Response 220
7.4 The Stiffness Method 224
7.5 Quick Solution Guide 230
7.6 References 235

Chapter 8 Static Analysis of Plane Frames 237
8.1 Getting Started 237
8.2 System Description 240
8.3 Solution by the Toolbox 243
8.3.1 System Setup 243
8.3.2 Response to External Forces 248
8.3.3 Response to Settlement of Supports 252
8.3.4 Total Response 256
8.3.5 Response of Frame Members 260
8.4 The Distributed Transfer Function Method 263
8.5 Quick Solution Guide 268
8.6 References 276

Part III Dynamics and Vibrations 277

Chapter 9 Dynamics of Particles and Rigid Bodies 279
9.1 Getting Started 279
9.2 Dynamics of Particles 280
9.2.1 Preliminaries 280
9.2.2 Kinematics 284
9.2.3 Kinetics of Single Particle 286
9.2.4 Particle Motion via Numerical Integration 289
9.2.5 Systems of Particles 297
9.2.6 Central Force Motion 303
9.3 Dynamics of Rigid Bodies in Plane Motion 307
9.3.1 Plane Motion 307
9.3.2 Mass Moments of Inertia 310
9.3.3 Kinetics 314
9.3.4 Simulation of Dynamic Response 317
9.4 Rigid Body Dynamics in Three Dimensions 323
9.4.1 Kinematics 323
9.4.2 Inertia Properties 327
9.4.3 Energy and Momentum 335

9.4.4 Equations of Motion 336
9.4.5 Rotation of Axisymmetric Bodies 340
9.5 Quick Solution Guide 349
9.6 References 350

Chapter 10 Vibration Analysis of One-Degree-of-Freedom Systems 351
10.1 Getting Started 351
10.2 System Description 353
10.3 Time Response 357
10.3.1 Free Response 357
10.3.2 Forced Response 361
10.3.3 Specific Forcing Functions 365
10.3.4 Total Response 374
10.3.5 Delayed Forcing Functions 376
10.3.6 Solution by Superposition 377
10.3.7 Mechanical Energy 379
10.3.8 Plotting Computed Response 382
10.4 Analytical Vibration Solutions 384
10.4.1 Analytical Methods 384
10.4.2 Solution by the Toolbox 389
10.5 Frequency Response 394
10.5.1 Harmonic Excitation 394
10.5.2 Base Excitation 400
10.5.3 Response under Rotating Unbalance 403
10.5.4 Vibration Isolation 406
10.6 Response to Periodic Excitation 410
10.6.1 Steady-State Solution in Fourier Series 410
10.6.2 Solution by the Toolbox 412
10.6.3 Fourier Coefficients of Certain Loads 419
10.7 Nonlinear Vibration 421
10.8 Quick Solution Guide 431
10.9 References 435

Chapter 11 Vibration and Control of Multiple-Degree-of-Freedom Systems 437
11.1 Getting Started 437
11.2 System Description 439
11.2.1 Introduction 439
11.2.2 Modes of Vibration 442
11.2.3 Solution by the Toolbox 444
11.3 Dynamic Response 450
11.3.1 Modal Analysis 450
11.3.2 Laplace Transform Method 458
11.3.3 Plotting Analytical Solutions 462
11.3.4 Runge-Kutta Algorithm 465
11.3.5 Solution by Superposition 468
11.3.6 Harmonic Excitation 472
11.4 Dynamic Vibration Absorption 475
11.4.1 Undamped Vibration Absorbers 476
11.4.2 Damped Vibration Absorbers 481

11.5 Transfer Function Formulation 487
11.5.1 Transfer Function and Green's Function 487
11.5.2 Open-Loop Transfer Function 491
11.6 Feedback Control 497
11.6.1 Vibration Control System 497
11.6.2 Position Control System 501
11.6.3 Solution by the Toolbox 506
11.7 Quick Solution Guide 515
11.8 References 518

Part IV Structural Dynamics 519

Chapter 12 Dynamics and Control of Euler-Bernoulli Beams 521
12.1 Getting Started 521
12.2 System Description 523
12.2.1 Governing Equation 523
12.2.2 Eigenvalue Problem 524
12.2.3 System Setup by the Toolbox 528
12.2.4 Animation of Modes of Vibration 534
12.2.5 Distributed Transfer Function Method 535
12.3 Dynamic Response 537
12.3.1 Modal Expansion 537
12.3.2 Specification of Damping 539
12.3.3 Free Vibration 540
12.3.4 Forced Vibration 544
12.3.5 Total Response 552
12.3.6 Animation of Time Response 554
12.3.7 Frequency Response 555
12.4 Feedback Control 559
12.4.1 Control System Formulation 559
12.4.2 Solution by the Toolbox 567
12.5 Dynamics and Control of Nonuniform Beams 581
12.5.1 Problem Statement 581
12.5.2 Rayleigh-Ritz Discretization 582
12.5.3 Modes of Vibration 588
12.5.4 Time Response 594
12.5.5 Control System Formulation 601
12.6 Quick Solution Guide 608
12.7 References 615

Chapter 13 Dynamic Analysis of Bars, Shafts, and Strings 617
13.1 Getting Started 617
13.2 System Description 619
13.2.1 Governing Equations 619
13.2.2 Eigenvalue Problem 622
13.2.3 System Setup by the Toolbox 624
13.2.4 Display of Modes of Vibration 627
13.3 Dynamic Response 630
13.3.1 Modal Expansion 631
13.3.2 Free Vibration 632

13.3.3 Forced Vibration 635
13.3.4 Total Response 639
13.3.5 Animation of Time Response 642
13.3.6 Frequency Response 643
13.4 Free Vibration of Stepped Systems 647
13.4.1 System Description 647
13.4.2 Free Vibration Analysis 651
13.4.3 Solution by the Toolbox 655
13.5 Quick Solution Guide 661
13.6 References 665

Chapter 14 Dynamic Analysis of Constrained, Combined, and Stepped Beams 667
14.1 Getting Started 667
14.2 Constrained Beams 670
14.2.1 System Description 670
14.2.2 System Setup and Eigensolutions 675
14.2.3 Eigenvalue Locus 683
14.2.4 Frequency Response 687
14.2.5 Multispan Beam Structures 689
14.2.6 Transient Response 691
14.3 Combined Beams 691
14.3.1 System Description 692
14.3.2 System Setup and Eigensolutions 696
14.3.3 Transient Response 698
14.3.4 Frequency Response 707
14.3.5 Oscillators for Vibration Absorption 712
14.4 Stepped Beams 715
14.4.1 Free Vibration Analysis 715
14.4.2 Solution by the Toolbox 721
14.5 Quick Solution Guide 729
14.6 References 736

Part V Two-Dimensional Elastic Continua 737

Chapter 15 Static Analysis of Linearly Elastic Bodies 739
15.1 Getting Started 739
15.2 Theory of Linear Elasticity 742
15.2.1 Stress and Strain 742
15.2.2 Basic Equations of Elasticity 749
15.2.3 Conversion of Elasticity Constants 753
15.2.4 Strain Energy 755
15.2.5 Principal of Minimum Potential Energy 755
15.3 Elasticity Problems in Two Dimensions 759
15.3.1 Plane Stress and Plane Strain 759
15.3.2 Governing Equations 760
15.3.3 Solution by Stress Function 761
15.3.4 Thermal Stresses 764
15.3.5 Elasticity Problems in Polar Coordinates 765
15.3.6 Stress Concentrations 769

15.4 Finite Element Method for 2-D Elasticity Problems 770
15.4.1 Finite Element Formulation 770
15.4.2 MATLAB Solutions in Rectangular Regions 777
15.4.3 MATLAB Solutions in Arbitrary-Shaped Regions 792
15.5 Quick Solution Guide 802
15.6 References 805

Chapter 16 Free Vibration of Membranes and Plates 807
16.1 Getting Started 807
16.2 Free Vibration of Membranes 809
16.2.1 Rectangular Membranes 810
16.2.2 Circular Membranes 816
16.3 Free Vibration of Rectangular Plates 823
16.3.1 Plate Theory 823
16.3.2 Eigenvalue Problem 827
16.3.3 Solution by the Toolbox 832
16.4 Free Vibration of Circular Plates 839
16.4.1 Equations in Polar Coordinates 839
16.4.2 Modes of Vibration 842
16.5 Quick Solution Guide 848
16.6 References 851

Appendix A *Commonly Used Mathematical Formulas* *853*
A.1 Algebraic Formulas 853
A.2 Areas and Volumes of Common Shapes 855
A.3 Trigonometry 857
A.4 Hyperbolic Functions 860
A.5 Derivatives and Integration 860
A.6 Series Expansion 862
A.7 Analytical Geometry 864
A.8 Vector Analysis 868
A.9 Matrix Theory 872
A.10 Complex Numbers and Complex Functions 878
A.11 Laplace Transforms 879
A.12 Inverse Laplace Transform via Partial Fraction Expansion 882

Appendix B *MATLAB Basics* *889*
B.1 Getting Started 889
B.2 Matrix and Vector Manipulations 893
B.3 Graphics 898
B.4 M-Files 901
B.5 Control Flow 903
B.6 Solution of Algebraic and Differential Equations 907
B.7 Control System Toolbox 910

Appendix C *The Distributed Transfer Function Method* *913*
C.1 DTFM for One-Dimensional Continua 913
C.2 Transfer Function Synthesis of Multibody Structures 917

C.3 DTFM for Two- and Three-Dimensional Problems 919
C.4 References 920

Appendix D *Conversion of Units* *923*

Appendix E *Mechanical Properties of Engineering Materials* *925*

Index *929*

Preface

This book is intended to supply engineering professionals and students with a comprehensive and definitive reference to statics and dynamics of solids and structures. The book is for use as a resource and design tool in research and development, and for use as a study guide and learning aid in engineering education.

The book is written to meet the needs for interactive computing in technical referencing and engineering education. These needs result from the design requirements for high-accuracy and high-performance structures, machines and devices in a variety of engineering applications, and from the trend of engineering curriculum development in response to today's environment of fast-changing technologies. Chapter 1 further explains the purpose and philosophy of this writing.

A unique feature of this book is the integration of the development of principles, formulas, and solutions with user-friendly interactive computer programs, written in the powerful and popular software MATLAB. These programs produce instant engineering solutions, which cover pages of contents contained in a conventional handbook, and beyond. Different from and complementary to general-purpose numerical codes, these programs deliver analytical results pertaining to many topics covered in the book. Furthermore, with the rich resources of MATLAB, these programs allow in-depth exploration of the physics of deformation, stress and motion by analysis, simulation, graphics, and animation.

This book contains five main parts: strength of materials, structural mechanics, dynamics and vibrations, structural dynamics, and two-dimensional elastic continua. Besides, the book presents feedback control of mechanical systems and flexible structures to a great extent. These subjects are covered in 15 chapters (Chapters 2 to 16), each being a self-contained package of subject review, fundamental theories, formulas, and a toolbox of MATLAB functions (computer programs) for numerical and analytical solutions. The MATLAB functions are stored in a CD-ROM that is attached to the book. In addition, Appendix A collects commonly used mathematical formulas for engineering analysis and Appendix B gives a tutorial on MATLAB-based computing and programming.

There are two important points regarding this book. First, the book keeps a good balance between fundamental theory and technical computation. Such a balance, in the author's opinion, will best serve design engineers who often need a quick subject review, and instant engineering solutions at the same time. Second, the organization of the text closely follows a curriculum on solid mechanics and structural dynamics adopted by a typical engineering school. This naturally renders the book a useful study guide and learning aid for many courses.

All the numerical results in the book are obtained through use of the attached MATLAB functions. All the artwork and tables in the book are created by the author; all the MATLAB functions of the book are generated by the author. For theories and formulas that are generally available, a list of references is provided at the end of each chapter. Specific formulas or results from certain resources have been duly acknowledged.

Although much effort has been made to avoid errors, some of them are bound to escape detection. The author welcomes and appreciates any comments and suggestions from readers for necessary corrections and for further improvement of the quality of this work.

Bingen Yang
Los Angeles, California
December 2004

Acknowledgements

I would like to express my appreciation to the staff at Elsevier, especially Joel Stein, Shoshanna Grossman, and Carl Soares, for their assistance, guidance, and high professional competence.

Certain results and formulas presented in this book are the outcome of my previous research partially sponsored by the National Science Foundation, the US Army Research Office, and NASA's Jet Propulsion Laboratory.

Finally, I wish to thank my wife Haiyan and my daughters Sonia and Tanya, for their love, dedication and encouragement. Without their endless support, this work could never have been completed.

This Page Intentionally Left Blank

1 Introduction

Inside
- The Making of This Book
- How to Use This Book

1.1 The Making of This Book

Computing is essentially important in both engineering education and engineering practice. The use of mathematical software packages can greatly enhance the learning of various topics in science and technology, and surely help increase the efficiency and accuracy of engineering designs. The recent technological advancements in computers, software, and telecommunications make it possible for individuals to enjoy an interactive and mobile computing environment in their study and work.

Taking advantage of such a fast-changing environment, this book is intended to provide a new type of reference to statics and dynamics of solids and structures, with unique interactive computing capabilities. The purpose, approach, scope, and features of this writing are described under the following headlines.

Who Will Find the Book Useful

This book is ideal for both professionals and students dealing with aerospace, mechanical, and civil engineering, as well as naval architecture, biomechanics, robotics, and mechtronics. For engineers and specialists, the book is a valuable resource and handy design tool in research and development. For engineering students at both undergraduate and graduate levels, the book serves as a useful study guide and powerful learning aid in many courses. And for instructors, the book offers an easy and efficient approach to curriculum development and teaching innovation.

Uniqueness

This book differs from standard handbooks in that it integrates the development of formulas, fundamental theories, mathematical models, and solution methods with user-friendly interactive computer programs. Unlike the commonly adopted approach of "finish-the-book-first-and-add-software-later," the text-software integration is harmonically fabricated in the

book writing from scratch. This unique merger of technical referencing and interactive computing allows instant solution of a variety of engineering problems and in-depth exploration of the physics of deformation, stress, and motion by analysis, simulation, graphics, and animation.

Interactive Computing with MATLAB

The computer programs for the book are written in the powerful and popular MATLAB, which is a premier software package that provides an interactive environment for technical computation. Different from many books that teach people *how to use* MATLAB in engineering analysis, this book shows *how to obtain* instant engineering solutions by hundreds of preprogrammed MATLAB functions from a CD-ROM that is attached to the book. These functions permit easy generation of data, figures, animation, and even analytical expressions, and produce results that are equivalent to the contents covered in pages of a conventional handbook and beyond.

Motive for Writing

This writing is motivated by the following two needs in engineering education and technical referencing.

Need for Interactive Computing Capabilities in Engineering Education The solution of a problem in an undergraduate engineering course usually requires knowledge in the following four areas:

- The background material of the problem in consideration, such as strength of materials, vibrations, and structural dynamics;
- Mathematical physics, including differential equations, linear algebra, and matrix theory;
- Solution algorithms, which can be either analytical or numerical; and
- Computer coding in programming languages like C, Fortran, and MATLAB.

While computer coding is normally introduced in the freshman year, adequate knowledge in mathematical physics and solution algorithms is not available until the senior year or later. This lack of mathematical skills often limits undergraduate teaching to a few "classroom problems." Of course, commercial computer programs may be used for solution of complicated problems. The usage of those codes, however, still requires a background in mathematical physics and numerical algorithms.

The current book fills this gap by offering adequate computing capabilities to many engineering courses. With this book, an undergraduate student in earlier years can solve various engineering problems without worrying about numerical algorithms. This allows the student to focus on important aspects of fundamental principles in engineering science and to explore the physical insight of practical problems. Moreover, with the interactive computing capabilities provided by this book, more advanced topics can be introduced to adapt an undergraduate or graduate curriculum to today's environment of emerging technologies.

Need for Interactive Computing Capabilities in Technical Referencing A standard reference collects formulas and tables that normally cover a number of simple cases. Although general-purpose computer codes are available, they usually only deliver numerical results and are not integrated with many analytical formulas given in a standard reference. Quite often, an engineer or specialist would like to get a quick solution for verifying a design concept or a research idea. In this case, a reference with attached computer programs, which yield numerical or analytical solutions according to user-selected parameters, boundary conditions, and loads, would definitely be desirable.

This book provides such needed interactive computing capabilities to technical referencing. With hundreds of preprogrammed MATLAB functions, numerical and analytical solutions of various engineering problems can be easily obtained. Besides facilitating quick concept proof in design and research, these solutions can serve as a benchmark for verification of numerical algorithms and computer codes developed by the user.

Scope

This book covers basic topics regarding solids and structures, including strength of materials, structural mechanics, elasticity, particle and rigid-body dynamics, vibrations, structural dynamics, and structural controls. As indicated by the table of contents, these topics are presented in five parts with a total of 15 chapters. Each chapter deals with a type of problem or a class of systems encountered in engineering. Each chapter is a self-contained package of subject review, fundamental theories, formulas, and a set (toolbox) of MATLAB functions for numerical and analytical solutions.

For efficient utility of this book, no attempt has been made to include every topic in such a wide range of subjects. Instead, the following three criteria have been applied in selecting the book materials:

(a) The problem in consideration is fundamentally important to engineering education and engineering practice. Examples include static analysis of Euler-Bernoulli beams (Chapter 2), stress and deformation of elastic bodies (Chapters 5 and 15), and rigid-body dynamics (Chapter 9).
(b) The problem in consideration is representative of a wide class of engineering applications. Examples include columns (Chapter 4), trusses (Chapter 7), frames (Chapter 8), and multispan beam structures (Chapters 6 and 14).
(c) The problem in consideration requires substantial analytical and numerical efforts for better understanding of its physics. Examples include vibration of multiple-degree-of-freedom systems (Chapter 11), dynamics and control of Euler-Bernoulli beams (Chapter 12), and vibration of plates (Chapter 16).

Special Features

Besides its unique interactive computing capabilities, this book has several special features, some of which are not available in the existing references.

New Formulas and Solutions This book contains many new formulas and analytical solutions. Examples include

- Analytical expressions of static response of beams subject to general external loads and arbitrary boundary disturbances;
- Exact static deflections and stresses of flexible frames under arbitrary external loads and support settlement;
- Influence lines of statically indeterminate multispan beam structures;
- Exact vibration solutions of one-degree-of-freedom systems subject to general forcing functions;
- Exact expressions of normalized mode shapes (eigenfunctions) of beams, bars, shafts, and strings, under arbitrary boundary conditions;
- Eigenfrequency loci of constrained beam structures;
- Control system formulation and design for beams with feedback controllers; and
- Exact free vibration solutions of plates with various boundary conditions.

What makes these new results more useful is that they can be obtained easily through use of the attached MATLAB toolboxes.

Exact Solution via the Distributed Transfer Function Method This book presents exact static and dynamic responses of beams, bars, shafts, columns, and frames, which are determined by the Distributed Transfer Function Method (DTFM). The DTFM is a closed-form analytical solution technique for modeling, analysis, and control of flexible structures. The DTFM is flexible in dealing with different geometric configurations and boundary conditions, and convenient in computer coding. The DTFM is introduced in Appendix C and its application to specific problems is given in related chapters.

Instant Animation of Motion and Vibration The MATLAB toolboxes of the book have functions for animating the modes of vibration and transient response of beams, bars, shafts, constrained and combined beam structures, and plates, the motion of rigid bodies in two and three dimensions, and the response of lumped parameter systems and flexible beams under feedback control. This animation functionality, which takes advantage of the rich resources of MATLAB, helps better understand the physics of motion and vibration and makes learning of difficult subjects a fun experience.

Integrated System Modeling and Controller Design Feedback control has wide applications in machines and structures. This important topic is addressed in the current book, for lumped dynamic systems (Chapter 11) and flexible beams (Chapter 12). For these systems, the book presents major steps in control system design, including system modeling, dynamic analysis, control system formulation, controller design and numerical simulation, and provides MATLAB functions for each of the steps. This integration of modeling, analysis, design, and simulation for feedback control of machines and structures is not available in any other reference on structural dynamics.

1.2 How to Use This Book

Chapters 2 to 16 cover topics in strength of materials, structural mechanics, elasticity, particle and rigid-body dynamics, vibrations, structural dynamics, and structural controls. Each of the chapters has the following parts:

- Getting started
- Fundamental principles, formulas, and solutions
- MATLAB functions
- Examples
- Quick Solution Guide
- References

Each chapter has a toolbox of MATLAB functions contained in the attached CD-ROM. In addition, Appendices A to E will be useful for engineering design and analyses.

Some key points in using this book are given below.

Getting Started

To start, find the right chapter from the Contents for the problem or system in consideration, then go to the first section of the chapter, titled *Getting Started.* This section tells what the chapter is about, how to install the MATLAB Toolbox, how to use the Toolbox through a tutorial example (in most chapters), and what to do next.

Fundamental Principles

The fundamental principles of each subject covered are briefly reviewed. Some derivations of theories and mathematical models are provided. For detailed information on these basic issues, a list of references is given at the end of each chapter.

Formulas and Solutions

Formulas and solutions in a few special cases can be directly found from the text of a chapter. Formulas and solutions in general cases of geometric configurations, boundary conditions, and loadings can be obtained through use of the MATLAB toolbox for the chapter. This requires computing and programming with MATLAB.

MATLAB

MATLAB is a software product of The MathWorks, Inc., headquartered in Natick, Massachusetts. MATLAB is a registered trademark of The MathWorks, Inc. To obtain MATLAB, visit the company's Web site http://www.mathworks.com/. A quick tutorial and brief summary of computing and programming with MATLAB is given in Appendix B of this book, which is useful for both beginners and advanced users.

MATLAB Functions in CD-ROM

Attached to this book is a CD-ROM with hundreds of preprogrammed MATLAB functions. These functions form 15 toolboxes, one for each of Chapters 2 to 16. The license agreement and limited warranty about the software package is given at the end of the book. For possible updated versions of the MATLAB toolboxes contained in the CD-ROM, check the publisher's Web site.

Windows

"Windows" are used to summarize the purpose and utility of the MATLAB functions from the attached CD-ROM. The windows are distributed in the text flow so that they are naturally related to the formulas and solutions presented.

Examples

The windows are normally followed by step-by-step examples demonstrating how the MATLAB functions can be used in analysis, simulation, graphics, and animation. Furthermore, each toolbox has a function `RunEx`, which, when launched, displays all the numerical examples contained in the chapter.

Quick Solution Guide

Each chapter has a section titled *Quick Solution Guide*. This section briefly describes the problem or system in consideration, lists the MATLAB functions from the toolbox with window or section numbers for easy reference, and outlines the solution procedure in the MATLAB-based computation. This section is especially convenient to those who are familiar with the material covered in the chapter and would like to engage in technical computation directly.

References

At the end of each chapter is a list of references for further reading. These references are mostly textbooks and monographs.

Unit Conversion

In all the examples, quantities are given in either the standard international system (SI) of units or nondimensional units. For conversion between SI system and the U.S. customary system, refer to Appendix D.

Mathematical Formulas

For convenience in engineering analyses, this book collects commonly used mathematical formulas in algebra, trigonometry, analytical geometry, calculus, vector analysis, matrix theory, complex analysis, differential equations, and Laplace transforms; see Appendix A.

Mechanical Properties of Engineering Materials

For convenience in engineering designs, the mechanical properties of selected engineering materials are given in Appendix E.

Comments and Technical Questions

For comments on this book and technical questions about the attached MATLAB toolboxes, please contact the author by the following mail and e-mail address:

Professor Bingen Yang
Department of Aerospace and Mechanical Engineering
University of Southern California
3650 McClintock Avenue, Room 430
Los Angeles, CA 90089-1453
E-mail: bingen@usc.edu

PART I
Strength of Materials

This Page Intentionally Left Blank

2 Static Analysis of Euler-Bernoulli Beams

Inside
- Getting Started
- Beam Theory
- Static Analysis by the Toolbox
- Moment of Inertia of Beam Cross-Section Area
- Quick Solution Guide
- References

2.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for static analysis of Euler-Bernoulli beams.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

Quick Tutorial

To show how to use the Toolbox, consider a simply-supported beam in Fig. 2.1.1, which has length $L = 1$ and bending stiffness $EI = 25$, and is subject to a pointwise force $f_0 = 1.2$ at its midpoint. In the MATLAB command window, type:

```
>> EI = 25;  L = 1; BC_Spec = [1 1];
>> setbeam(EI, L, BC_Spec)
>> Load_Spec = [0  0.5  1.2];
>> y = beamf(Load_Spec);
>> plotbeam(y)
```

where >> is the prompt in the MATLAB command window. This yields the spatial distributions of the displacement (transverse deflection), rotation, bending moment, and shear force of the beam as plotted in Fig. 2.1.2, and the maximum response and the reactions as shown below.

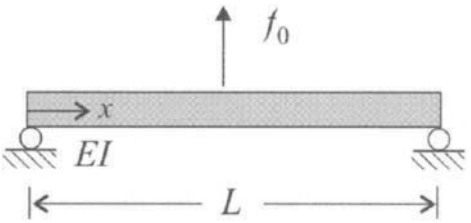


FIGURE 2.1.1 A simply-supported beam under a pointwise force.

```
Maximum Beam Response in Absolute Value
---------------------------------------
 Max displacement = 0.001,  location x = 0.5
 Max rotation (degrees) = 0.17189, location x = 0
 Max bending moment = -0.3,  location x = 0.5
 Max shear force = -0.6,  location x = 0
```

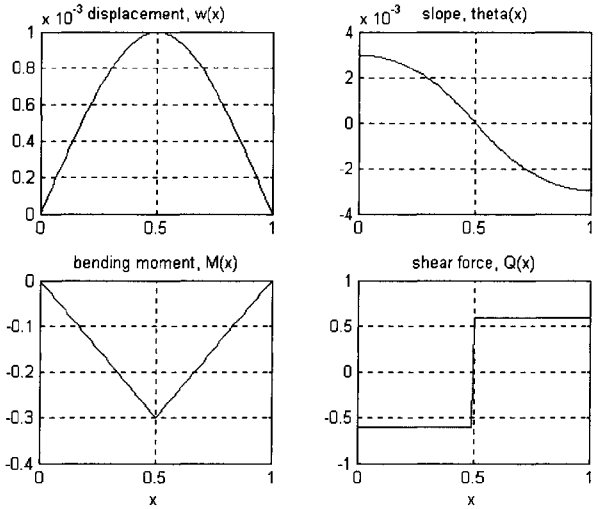


FIGURE 2.1.2 Beam response distribution.

```
Reactions at Two Ends of the Beam
---------------------------------
Sign convention for support reactions:
* Positive moment Mc: counterclockwise
* Positive force Rc: upward

At left boundary (x = 0):   Mc = 0,  Rc = -0.6
At right boundary (x = L):  Mc = 0,  Rc = -0.6
```

Also, the command

```
>> mathf(Load_Spec)
```

produces the analytical expressions of the beam response as follows

```
Beam response for 0 <= x <= 0.5:
 Displacement    w(x) = -0.004*x^3 + (0)*x^2 + (0.003)*x + (0)
 Rotation    dw(x)/dx = -0.012*x^2 + (0)*x + (0.003)
 Bending moment  M(x) = -0.6*x + (0)
 Shear force     Q(x) = -0.6

Beam response for 0.5 <= x <= 1:
 Displacement    w(x) = 0.004*x^3 + (-0.012)*x^2 + (0.009)*x + (-0.001)
 Rotation    dw(x)/dx = 0.012*x^2 + (-0.024)*x + (0.009)
 Bending moment  M(x) = 0.6*x + (-0.6)
 Shear force     Q(x) = 0.6
```

In the above static analysis, four functions from the Toolbox have been used:

(a) Function **setbeam** that assigns beam parameters and sets up beam boundary conditions;
(b) Function **beamf** that computes beam response under an external force and puts the computed results in a matrix y;
(c) Function **plotbeam** that plots the computed beam response versus the spatial coordinate x and shows the maximum beam response and reaction forces at beam supports; and
(d) Function **mathf** that delivers the analytical expressions of beam response.

These functions, along with others from the Toolbox, will be described in sequel.

For a quick solution, the reader may refer directly to the Quick Solution Guide (Section 2.5), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To understand better how the toolbox works, the reader is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 2.6.

2.2 Beam Theory

2.2.1 Static Problem

Governing Equation The transverse displacement (deflection) $w(x)$ of a uniform Euler-Bernoulli beam under static loads, as shown in Fig. 2.2.1, is governed by the differential

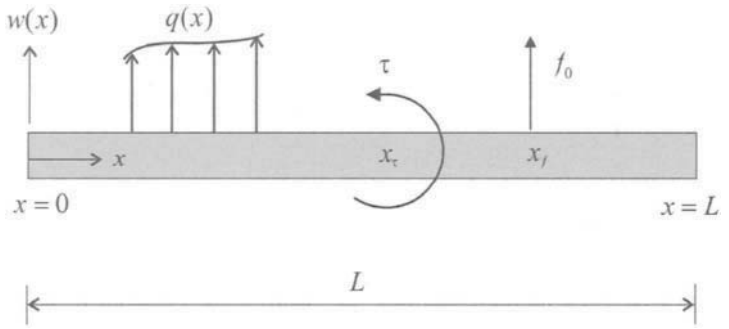


FIGURE 2.2.1 A uniform Euler-Bernoulli beam under static loads.

equation (Reference 2)

$$EI\frac{d^4}{dx^4}w(x) = f(x), \quad 0 \le x \le L \tag{2.1}$$

where EI and L are the bending stiffness and length of the beam, respectively. The forcing function for the loads shown in Fig. 2.2.1 has the form

$$f(x) = q(x) + f_0\delta(x - x_f) - \tau\frac{d}{dx}\delta(x - x_\tau)$$

where $q(x)$ is a distributed external load, f_0 is a pointwise force applied at x_f, τ is a torque applied at x_τ, and $\delta(\cdot)$ is the Dirac delta function. Refer to Section 2.2.3 for the derivation of Eq. (2.1). The sign convention for beam displacement and forces is as follows. The positive direction of beam displacement and transverse forces (either pointwise or distributed) is upward (↑); the positive direction of a torque is counterclockwise (↺).

Boundary Conditions The boundary conditions of the beam can be written as

$$\begin{aligned} &\text{At the left end} && B_{01}[w(x)]_{x=0} = \alpha_1, \quad B_{02}[w(x)]_{x=0} = \alpha_2 \\ &\text{At the right end} && B_{L1}[w(x)]_{x=L} = \beta_1, \quad B_{L2}[w(x)]_{x=L} = \beta_2 \end{aligned} \tag{2.2}$$

where B_{01}, B_{02}, B_{L1}, and B_{L2} are spatial differential operators, and α_i and β_i represent boundary disturbances. Seven types of beam boundary conditions with associate boundary disturbances are given in Table 2.2.1. In the table, u_0 and u_L are transverse boundary displacements, θ_0 and θ_L rotations, M_0 and M_L moments, and Q_0 and Q_L transverse forces. The positive direction of boundary displacements and transverse forces is upward; the positive direction of boundary rotations and moments is counterclockwise.

Static Response The static response of a beam includes:

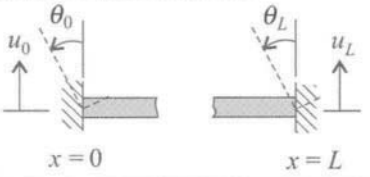
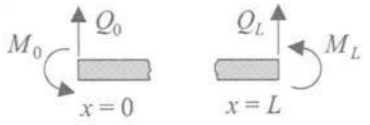
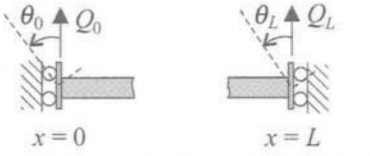
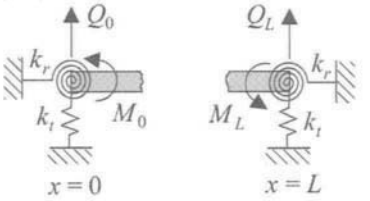
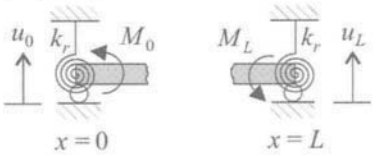
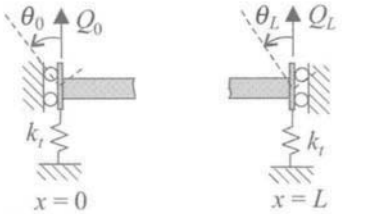
Displacement (transverse deflection) $w(x)$
Rotation or slope

$$\theta(x) = \frac{d}{dx}w(x) \tag{2.3}$$

Bending moment

$$M(x) = EI\frac{d^2}{dx^2}w(x) \tag{2.4}$$

TABLE 2.2.1 Boundary Conditions and Boundary Disturbances

Boundary Type	*Boundary Conditions*
(B1) Pinned or hinged end	Left end: $w(0)=u_0,\quad -EI\dfrac{d^2w(0)}{dx^2}=M_0$ Right end: $w(L)=u_L,\quad EI\dfrac{d^2w(L)}{dx^2}=M_L$
(B2) Clamped or fixed end 	Left end: $w(0)=u_0,\quad \dfrac{dw(0)}{dx}=\theta_0$ Right end: $w(L)=u_L,\quad \dfrac{dw(L)}{dx}=\theta_L$
(B3) Free end 	Left end: $-EI\dfrac{d^2w(0)}{dx^2}=M_0,\quad EI\dfrac{d^3w(0)}{dx^3}=Q_0$ Right end: $EI\dfrac{d^2w(L)}{dx^2}=M_L,\quad -EI\dfrac{d^3w(L)}{dx^3}=Q_L$
(B4) Sliding or guided end 	Left end: $\dfrac{dw(0)}{dx}=\theta_0,\quad EI\dfrac{d^3w(0)}{dx^3}=Q_0$ Right end: $\dfrac{dw(L)}{dx}=\theta_L,\quad -EI\dfrac{d^3w(L)}{dx^3}=Q_L$
(B5) Elastic end 	Left end: $k_r\dfrac{dw(0)}{dx}-EI\dfrac{d^2w(0)}{dx^2}=M_0,\quad k_tw(0)+EI\dfrac{d^3w(0)}{dx^3}=Q_0$ Right end: $k_r\dfrac{dw(L)}{dx}+EI\dfrac{d^2w(L)}{dx^2}=M_L,\quad k_tw(L)-EI\dfrac{d^3w(L)}{dx^3}=Q_L$
(B6) Hinged-elastic end 	Left end: $w(0)=u_0,\quad k_r\dfrac{dw(0)}{dx}-EI\dfrac{d^2w(0)}{dx^2}=M_0$ Right end: $w(L)=u_L,\quad k_r\dfrac{dw(L)}{dx}+EI\dfrac{d^2w(L)}{dx^2}=M_L$
(B7) Sliding-elastic end 	Left end: $\dfrac{dw(0)}{dx}=\theta_0,\quad k_tw(0)+EI\dfrac{d^3w(0)}{dx^3}=Q_0$ Right end: $\dfrac{dw(L)}{dx}=\theta_L,\quad k_tw(L)-EI\dfrac{d^3w(L)}{dx^3}=Q_L$

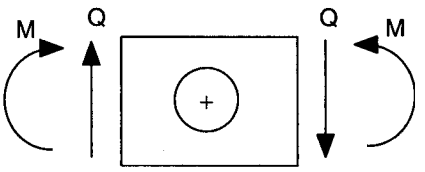


FIGURE 2.2.2 Sign convention of bending moment and shear force.

Shear force

$$Q(x) = EI\frac{d^3}{dx^3}w(x) \tag{2.5}$$

Here M and Q are called the internal forces of the beam. The positive direction of $w(x)$ is upward; the positive direction of rotation is counterclockwise; and the sign convention of bending moment and shear force is given in Fig. 2.2.2.

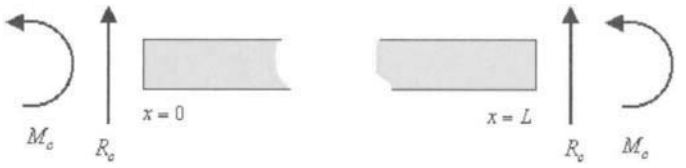


FIGURE 2.2.3 Sign convention of reactions.

Reactions at Supports All the boundaries listed in Table 2.2.1, except for B3 (free ends), exert reaction moments and/or reaction forces to the beam when it is subject to external loads. These reactions are to support the beam by balancing the external forces. For this reason, these boundaries are also called *supports*. The reactions are represented by the following equations:

$$\begin{array}{ll} \text{At the left end } (x = 0) & M_c = -EIw''(0), \quad R_c = EIw'''(0) \\ \text{At the right end } (x = L) & M_c = EIw''(L), \quad R_c = -EIw'''(L) \end{array}$$

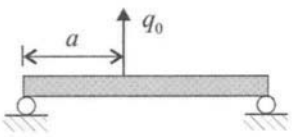
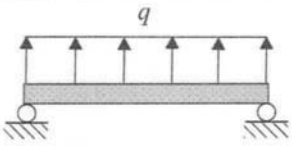
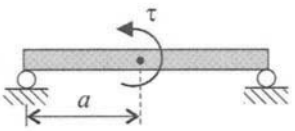
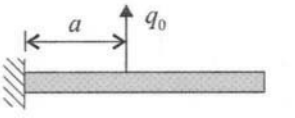
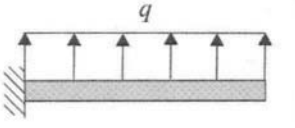
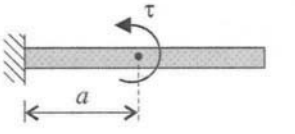
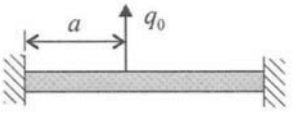
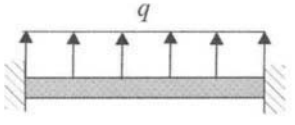
where $w' = dw/dx$. The sign convention for the reactions is the same as the boundary disturbances (see Fig. 2.2.3); that is,

- A positive reaction moment (M_c) is in the counterclockwise direction; and
- A positive reaction force (R_c) is in the upward direction.

A fundamental problem in static analysis of Euler-Bernoulli beams is stated as follows: *Given external loads and boundary disturbances, determine the beam response (displacement, rotation, bending moment, and shear force) that is governed by Eq. (2.1) and the boundary conditions (2.2).*

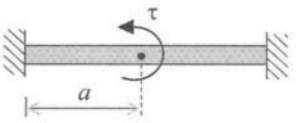
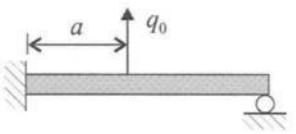
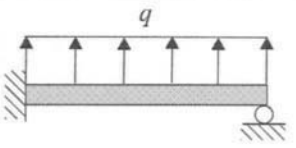
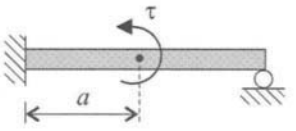
Listed in Table 2.2.2 is the static beam deflection in some cases of boundary conditions and loads. The bending moment and shear force in each case can be derived according to Eqs. (2.4) and (2.5). The deflection solutions in the table can be obtained by the methods given in Section 2.2.2. Also, refer to Section 2.3.5 for analytical solutions with general beam configurations and arbitrary external loads.

TABLE 2.2.2 Static Beam Deflection

Beam Type and Load	*Transverse Deflection*
	$w(x) = \frac{1}{6EI} R_0 x^3 + \theta_0 x \qquad \text{for } 0 \le x < a$ $w(x) = \frac{1}{6EI} R_0 x^3 + \theta_0 x + \frac{q_0}{6EI} (x-a)^3 \qquad \text{for } a \le x \le L$ $\text{with } R_0 = -q_0 \mu, \ \theta_0 = -\frac{q_0 L^2}{6EI} \left(\mu^3 + \mu\right), \ \mu = 1 - a/L$
	$w(x) = \frac{q}{24EI} \left(x^4 - 2Lx^3 + L^3 x\right) \qquad \text{for } 0 \le x \le L$
	$w(x) = \frac{1}{6EI} R_0 x^3 + \theta_0 x \qquad \text{for } 0 \le x < a$ $w(x) = \frac{1}{6EI} R_0 x^3 + \theta_0 x - \frac{\tau}{2EI} (x-a)^2 \qquad \text{for } a \le x \le L$ $\text{with } R_0 = \frac{\tau}{L}, \ \theta_0 = \frac{\tau L}{6EI} \left(3\mu^2 - 1\right), \ \mu = 1 - a/L$
	$w(x) = \frac{q_0}{6EI} \left(-x^3 + 3ax^2\right) \qquad \text{for } 0 \le x < a$ $w(x) = \frac{q_0}{6EI} a^2 (3x - a) \qquad \text{for } a \le x \le L$
	$w(x) = \frac{q}{24EI} \left(x^4 - 4Lx^3 + 6L^2 x^2\right) \qquad \text{for } 0 \le x \le L$
	$w(x) = \frac{\tau}{2EI} x^2 \qquad \text{for } 0 \le x < a$ $w(x) = \frac{\tau a}{2EI} (2x - a) \qquad \text{for } a \le x \le L$
	$w(x) = \frac{1}{EI} \left(\frac{1}{6} R_0 x^3 - \frac{1}{2} M_0 x^2\right) \qquad \text{for } 0 \le x < a$ $w(x) = \frac{1}{EI} \left(\frac{1}{6} R_0 x^3 - \frac{1}{2} M_0 x^2 + \frac{q_0}{6} (x-a)^3\right) \qquad \text{for } a \le x \le L$ $\text{with } R_0 = q_0 \left(2\mu^3 - 3\mu^2\right), \ M_0 = q_0 L \left(\mu^3 - \mu^2\right), \ \mu = 1 - a/L$
	$w(x) = \frac{q}{24EI} \left(x^4 - 2Lx^3 + L^2 x^2\right) \qquad \text{for } 0 \le x \le L$

(continued)

TABLE 2.2.2 Static Beam Deflection (*Continued*)

Beam Type and Load	Transverse Deflection
	$w(x) = \frac{1}{EI}\left(\frac{1}{6}R_0x^3 - \frac{1}{2}M_0x^2\right)$ for $0 \le x < a$ $w(x) = \frac{1}{EI}\left(\frac{1}{6}R_0x^3 - \frac{1}{2}M_0x^2 - \frac{\tau}{2}(x-a)^2\right)$ for $a \le x \le L$ with $R_0 = \frac{3\tau}{2L}\left(2\mu - \mu^2\right)$, $M_0 = \frac{\tau}{2}\left(2 - 3\mu^2\right)$, $\mu = 1 - a/L$
	$w(x) = \frac{1}{EI}\left(\frac{1}{6}R_0x^3 - \frac{1}{2}M_0x^2\right)$ for $0 \le x < a$ $w(x) = \frac{1}{EI}\left(\frac{1}{6}R_0x^3 - \frac{1}{2}M_0x^2 + \frac{q_0}{6}(x-a)^3\right)$ for $a \le x \le L$ with $R_0 = \frac{q_0}{2}\left(\mu^3 - 3\mu\right)$, $M_0 = \frac{q_0L}{2}\left(\mu^3 - \mu\right)$, $\mu = 1 - a/L$
	$w(x) = \frac{q}{48EI}\left(2x^4 - 5Lx^3 + \frac{1}{16}L^2x^2\right)$ for $0 \le x \le L$
	$w(x) = \frac{1}{EI}\left(\frac{1}{6}R_0x^3 - \frac{1}{2}M_0x^2\right)$ for $0 \le x < a$ $w(x) = \frac{1}{EI}\left(\frac{1}{6}R_0x^3 - \frac{1}{2}M_0x^2 - \frac{\tau}{2}(x-a)^2\right)$ for $a \le x \le L$ with $R_0 = \frac{3\tau}{2L}\left(1 - \mu^2\right)$, $M_0 = \frac{\tau}{2}\left(1 - 3\mu^2\right)$, $\mu = 1 - a/L$

In the next two subsections, three solution methods for beam static analysis are presented and the fundamentals of the Euler-Bernoulli beam in bending are reviewed. The reader who is only interested in using the MATLAB toolbox can directly move to Section 2.3.

2.2.2 Solution Methods

Many solution methods are available for static analysis of Euler-Bernoulli beams. This section introduces three analytical solution methods for the beam problem, namely the method of singularity functions, the boundary value approach, and the distributed transfer function method.

Method of Singularity Functions This method is often presented in textbooks on strength of materials or solid mechanics; see References 3 and 4 for instance. In the method, Eq. (2.1) is integrated four times, yielding

$$EIw(x) = \frac{1}{6}R_0x^3 - \frac{1}{2}M_0x^2 + EI(\theta_0 x + w_0) + \int_0^x \int_0^x \int_0^x \int_0^x f(x)\,dx^4 \tag{2.6}$$

where the initial parameters

$$w_0 = w(0), \quad \theta_0 = w'(0), \quad M_0 = -EIw''(0), \quad R_0 = EIw'''(0) \tag{2.7}$$

The sign convention of R_0 and M_0 follows that in Fig. 2.2.3. Physically, R_0 and M_0 can be known boundary disturbances, or unknown support reactions.

The integral in Eq. (2.6) can be expressed by the singularity function $\langle x - x_0\rangle^n$.

Definition: The singularity function $\langle x - x_0\rangle^n$ is defined as follows:
for $n \geq 0$

$$\langle x - x_0\rangle^n = \begin{cases} 0 & \text{for } x < x_0 \\ (x - x_0)^n & \text{for } x \geq x_0 \end{cases} \tag{2.8}$$

for $n = -1, -2$

$$\langle x - x_0\rangle^n = 0 \quad \text{for } x \neq x_0 \tag{2.9a}$$

and

$$\int_0^x \langle x - x_0\rangle^n dx = \langle x - x_0\rangle^{n+1} \tag{2.9b}$$

From the above definition, $\langle x - x_0\rangle^{-1}$ actually is the delta function $\delta(x - x_0)$, and $\langle x - x_0\rangle^0$ is the unite step function.

With the singularity function, the integral $\int_0^x \int_0^x \int_0^x \int_0^x f(x)\, dx^4$ for several types of external forces is given in Table 2.2.3. The solution of a beam problem can be obtained according to Eq. (2.6) and Table 2.2.3, as shown in the following example.

EXAMPLE 2.1

Consider a clamped-pinned beam subject to a torque at the midpoint; see Fig. 2.2.4. The beam response, by Eq. (2.6) and Table 2.2.3 is

$$EIw(x) = \frac{1}{6}R_0x^3 - \frac{1}{2}M_0x^2 + EI(\theta_0 x + w_0) - \frac{\tau}{2}\langle x - L/2\rangle^2$$

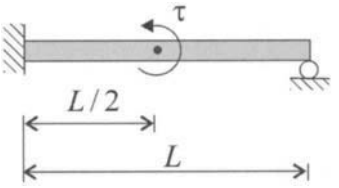
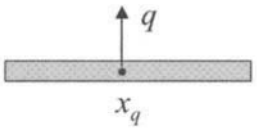
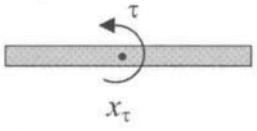
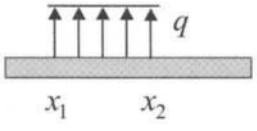
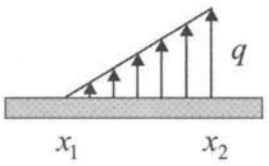


FIGURE 2.2.4

TABLE 2.2.3 External Forces and Integrals

External Force	*Integral*
Pointwise force 	$f(x) = q\langle x - x_q\rangle^{-1}$ $\int_0^x \int_0^x \int_0^x \int_0^x f(x)\,dx^4 = \frac{q}{6}\langle x - x_q\rangle^3$
Torque 	$f(x) = -\tau\langle x - x_\tau\rangle^{-2}$ $\int_0^x \int_0^x \int_0^x \int_0^x f(x)\,dx^4 = -\frac{\tau}{2}\langle x - x_\tau\rangle^2$
Uniform force 	$f(x) = q\langle x - x_1\rangle^0 - q\langle x - x_2\rangle^0$ $\int_0^x \int_0^x \int_0^x \int_0^x f(x)\,dx^4 = \frac{q}{24}\langle x - x_1\rangle^4 - \frac{q}{24}\langle x - x_2\rangle^4$
Linearly distributed load 	$f(x) = k\langle x - x_1\rangle^1 - k\langle x - x_2\rangle^1 - q\langle x - x_2\rangle^0$ where $k = q/(x_2 - x_1)$ $\int_0^x \int_0^x \int_0^x \int_0^x f(x)\,dx^4 = \frac{k}{120}\langle x - x_1\rangle^5 - \frac{k}{120}\langle x - x_2\rangle^5 - \frac{q}{24}\langle x - x_2\rangle^4$

By the boundary conditions

$$w(0) = 0, \quad w'(0) = 0, \quad w(L) = 0, \quad w''(L) = 0$$

the initial parameters of the beam are determined as

$$w_0 = 0, \quad \theta_0 = 0, \quad M_0 = \frac{\tau}{8}, \quad R_0 = \frac{9\tau}{8L}.$$

Thus, the deflection of the beam is

$$w(x) = \frac{\tau}{EI}\left(\frac{3}{16L}x^3 - \frac{1}{16}x^2 - \frac{1}{2}\langle x - L/2\rangle^2\right)$$

and the rotation, bending moment, and shear force are

$$\theta(x) = \frac{\tau}{EI}\left(\frac{9}{16L}x^2 - \frac{1}{8}x - \langle x - L/2\rangle^1\right)$$

$$M(x) = \tau\left(\frac{9}{8L}x - \frac{1}{8} - \langle x - L/2\rangle^0\right)$$

$$Q(x) = \frac{9\tau}{8L}$$

The beam response is plotted in Fig. 2.2.5 for $EI = 1$, $L = 1$, and $\tau = 1$.

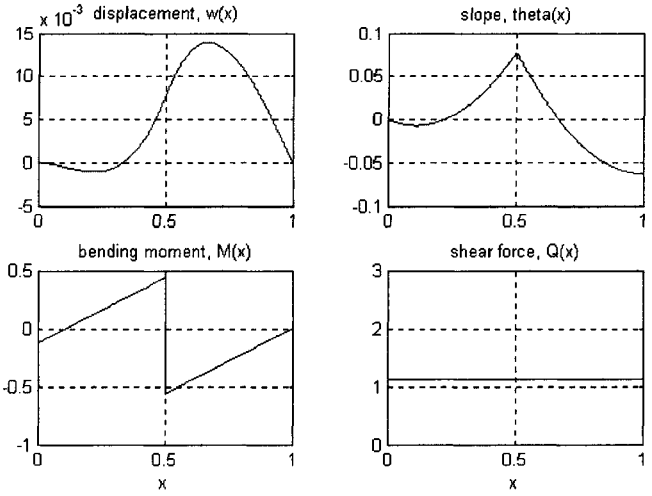


FIGURE 2.2.5

Boundary Value Approach

Response to boundary disturbances The response of a beam under boundary disturbances is governed by

$$EI\frac{d^4}{dx^4}w(x) = 0, \quad 0 \le x \le L \tag{2.10}$$

subject to boundary conditions as described in Table 2.2.1. The solution of the above equation is of the form

$$w(x) = \gamma_0 + \gamma_1 x + \gamma_2 x^2 + \gamma_3 x^3 \tag{2.11}$$

where γ_0, γ_1, γ_2, and γ_3 are coefficients to be determined by the beam boundary conditions.

Response to a pointwise force The response of a beam under a pointwise force q at $x = x_f$ is governed by

$$EI\frac{d^4}{dx^4}w(x) = q\delta(x - x_f), \qquad 0 \le x \le L \tag{2.12}$$

The solution of Eq. (2.12) is

$$\begin{aligned} w_1(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 \qquad \text{for } 0 \le x \le x_f \\ w_2(x) &= b_0 + b_1 x + b_2 x^2 + b_3 x^3 \qquad \text{for } x_f \le x \le L \end{aligned} \tag{2.13}$$

where the coefficients a_j and b_k are determined by the boundary conditions of the beam and the compatibility conditions

$$\begin{aligned} w_1(x_f) &= w_2(x_f), \qquad w_1'(x_f) = w_2'(x_f) \\ w_1''(x_f) &= w_2''(x_f), \qquad w_1'''(x_f) + \frac{q}{EI} = w_2'''(x_f) \end{aligned} \tag{2.14}$$

where $w' = dw/dx$.

Response to a torque The response of a beam under a torque τ at $x = x_\tau$ is governed by

$$EI\frac{d^4}{dx^4}w(x) = -\tau\frac{d}{dx}\delta(x - x_\tau), \qquad 0 \le x \le L \tag{2.15}$$

The solution of Eq. (2.15) is

$$\begin{aligned} w_1(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 \qquad \text{for } 0 \le x \le x_\tau \\ w_2(x) &= b_0 + b_1 x + b_2 x^2 + b_3 x^3 \qquad \text{for } x_\tau \le x \le L \end{aligned} \tag{2.16}$$

where the coefficients a_j and b_k are determined by the beam boundary conditions and the following compatibility conditions

$$\begin{aligned} w_1(x_\tau) &= w_2(x_\tau), \qquad w_1'(x_\tau) = w_2'(x_\tau) \\ w_1''(x_\tau) - \frac{\tau}{EI} &= w_2''(x_\tau), \qquad w_1'''(x_\tau) = w_2'''(x_\tau) \end{aligned} \tag{2.17}$$

Response to distributed forces Consider two types of distributed forces applied in the region $x_1 \le x \le x_2$:

(a) A force of polynomial distribution

$$f_1(x) = \begin{cases} c_0 + c_1 x + c_2 x^2, & \text{for } x_1 \le x \le x_2 \\ 0, & \text{otherwise} \end{cases} \tag{2.18a}$$

(b) A force of sinusoidal distribution

$$f_2(x) = \begin{cases} A\sin(\omega x + \phi), & \text{for } x_1 \le x \le x_2 \\ 0, & \text{otherwise} \end{cases} \tag{2.18b}$$

where $0 \le x_1 < x_2 \le L$. The static displacement of a beam under $f_1(x)$ or $f_2(x)$ is described by

$$EI\frac{d^4}{dx^4}w(x) = f_k(x), \quad k = 1 \text{ or } 2 \tag{2.19}$$

The solution of Eq. (2.19) is of the form

$$\begin{aligned} w_1(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 \qquad \text{for } 0 \le x \le x_1 \\ w_2(x) &= d_0 + d_1 x + d_2 x^2 + d_3 x^3 + w_p(x) \quad \text{for } x_1 \le x \le x_2 \\ w_3(x) &= b_0 + b_1 x + b_2 x^2 + b_3 x^3 \qquad \text{for } x_2 \le x \le L \end{aligned} \tag{2.20}$$

where the coefficients a_j, b_k, and d_l are determined by the boundary conditions of the beam and the compatibility conditions

$$\begin{aligned} &w_1(x_1) = w_2(x_1), \quad w_1'(x_1) = w_2'(x_1), \quad w_1''(x_1) = w_2''(x_1), \quad w_1'''(x_1) = w_2'''(x_1) \\ &w_2(x_2) = w_3(x_2), \quad w_2'(x_2) = w_3'(x_2), \quad w_2''(x_2) = w_3''(x_2), \quad w_2'''(x_2) = w_3'''(x_2) \end{aligned} \tag{2.21}$$

and $w_p(x)$ is a particular solution of Eq. (2.19). For force $f_1(x)$ (polynomial distribution), a particular solution is

$$w_{p1}(x) = \frac{1}{EI}\left(\frac{1}{24}c_0 x^4 + \frac{1}{120}c_1 x^5 + \frac{1}{360}c_2 x^6\right), \quad \text{for } x_1 \le x \le x_2 \tag{2.22a}$$

For force $f_2(x)$ (sinusoidal distribution), a particular solution is

$$w_{p2}(x) = \frac{A}{EI\omega^4}\sin(\omega x + \phi), \quad \text{for } x_1 \le x \le x_2 \tag{2.22b}$$

Once the coefficients γ_i, a_j, b_k, and d_l are determined by the boundary and compatibility conditions, the beam response is given in the exact form Eq. (2.20).

Distributed Transfer Function Method The MATLAB functions of this chapter are developed based on the Distributed Transfer Function Method (DTFM), which is an exact analytical modeling and solution method for many one-dimensional distributed systems including bars, shafts, strings, and beams. Refer to Appendix C for more information on the DTFM.

Spatial state formulation In the DTFM, Eq. (2.1) is first cast into an equivalent spatial state form

$$\frac{d}{dx}\{\eta(x)\} = [F]\,\eta(x) + \frac{1}{EI}f(x)\cdot\{e_4\}, \quad 0 \le x \le L \tag{2.23}$$

where

$$\{\eta(x)\} = \begin{pmatrix} w(x) \\ w'(x) \\ w''(x) \\ w'''(x) \end{pmatrix}, \quad [F] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \{e_4\} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.24}$$

The boundary conditions can also be written in terms of $\{\eta\}$:

$$[M_b]\{\eta(0)\} + [N_b]\{\eta(L)\} = \{\gamma_b\} \tag{2.25}$$

where $[M_b]$ and $[N_b]$ are four-by-four boundary matrices and $\{\gamma_b\}$ is a boundary disturbance vector. For instance, for a clamped-free (cantilever) beam subject to an end moment M_L, the

boundary conditions are

$$\begin{aligned} &\text{Left end:} && w(0) = 0, \quad w'(0) = 0 \\ &\text{Right end:} && EIw''(L) = M_L, \quad w'''(L) = 0 \end{aligned}$$

which are converted to Eq. (2.25) with

$$[M_b] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad [N_b] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & EI & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \{\gamma_b\} = \begin{pmatrix} 0 \\ 0 \\ M_L \\ 0 \end{pmatrix}$$

Hence, the original beam problem is reduced to the state equation (2.23) subject to the boundary condition (2.25).

Solution by distributed transfer functions By the DTFM, the solution of Eq. (2.23) with the boundary condition (2.25) is

$$\eta(x) = \frac{1}{EI} \int_0^L [G(x,\xi)] f(\xi)\, d\xi \, \{e_4\} + [H(x)]\, \{\gamma_b\} \tag{2.26}$$

where the Green's function $[G(x,\xi)]$ and the boundary influence function $[H(x)]$ are

$$\begin{aligned} [G(x,\xi)] &= \begin{cases} [H(x)]\,[M_b]\, e^{-[F]\xi}, & x \geq \xi \\ -[H(x)]\,[N_b]\, e^{[F](L-\xi)}, & x \leq \xi \end{cases} \\ [H(x)] &= e^{[F]x} \left([M_b] + [N_b]\, e^{[F]L} \right)^{-1} \end{aligned} \tag{2.27}$$

and $e^{[F]x}$ is an exponential matrix of the form

$$e^{[F]x} = \begin{bmatrix} 1 & x & x^2/2 & x^3/6 \\ 0 & 1 & x & x^2/2 \\ 0 & 0 & 1 & x \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

In the DTFM, $[G(x,\xi)]$ and $[H(x)]$ are called *distributed transfer functions*. The static displacement and internal forces of the beam are expressed as

$$\begin{pmatrix} w(x) \\ M(x) \\ Q(x) \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & EI & 0 \\ 0 & 0 & 0 & EI \end{bmatrix} \eta(x) \tag{2.29}$$

By Eqs. (2.26) and (2.27), the beam response is expressed as

$$\eta(x) = \frac{1}{EI}[H(x)]\left\{[M_b]\int_0^x e^{-[F]\xi} f(\xi)\,d\xi - [N_b]\,e^{[F]L}\int_x^L e^{-[F]\xi} f(\xi)\,d\xi\right\}\{e_4\} + [H(x)]\{\gamma_b\} \tag{2.30}$$

This equation involves the integrals $\int \xi^j f(\xi)\,d\xi$ for $j = 0, 1, 2, 3$, which can be evaluated by exact quadrature. Therefore, the beam solution by the DTFM is in exact and closed form.

EXAMPLE 2.2

A beam of stiffness EI and length L is subject to a uniform load in its entire domain, i.e., $f(x) = q_0$ for $0 \le x \le L$. The response of the beam by Eq. (2.30) is

$$\eta(x) = \frac{q_0}{EI}[H(x)]\left\{[M_b]\int_0^x e^{-[F]\xi}\,d\xi - [N_b]\,e^{[F]L}\int_x^L e^{-[F]\xi}\,d\xi\right\}\{e_4\}.$$

By Eq. (2.28), it is easy to show that

$$\Delta(\xi) \equiv \int e^{-[F]\xi}\,d\xi\,\{e_4\} = \left(-\frac{1}{24}\xi^4 \quad \frac{1}{6}\xi^3 \quad -\frac{1}{2}\xi^2 \quad \xi\right)^T.$$

It follows that the response of the beam is in the exact and closed form

$$\eta(x) = \frac{q_0}{EI}[H(x)]\left\{[M_b]\,\Delta(x) - [N_b]\,e^{[F]L}\,(\Delta(L) - \Delta(x))\right\}.$$

The above solution is valid for the beam with arbitrary boundary conditions, provided that the boundary matrices $[M_b]$ and $[N_b]$ are properly assigned.

2.2.3 Fundamentals of Euler-Bernoulli Beams

In this section, the fundamentals of the Euler-Bernoulli beam in bending are reviewed to show how Eq. (2.1) is derived. For more information on the beam theory, refer to any of the references listed in Section 2.6.

Basic Assumptions of Euler-Bernoulli Beams

(a) Kinematic assumption (Euler-Bernoulli hypothesis): Plane sections of a beam normal to its axis (neutral axis) remain planes after the beam experiences bending deformation.

(b) Linear elastic assumption: In deformation, the normal stress and strain of beam in the longitudinal direction satisfy the Hooke's law:

$$\sigma_x = E\varepsilon_x \tag{2.31}$$

where E is Young's modulus.

(c) Small deflection assumption: The beam undergoes small deformation such that the rotation w' is a negligible quantity in comparison with unity.

(d) The beam is in pure bending.

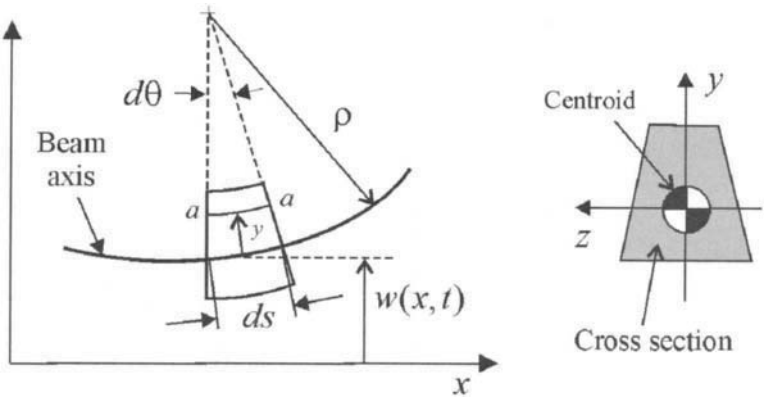


FIGURE 2.2.6 A beam segment in bending.

Curvature of Beam Axis Under Assumption (a), the normal strain of an infinitesimal fiber segment *a-a* of the beam (see Fig. 2.2.6) that is a distance y from the beam is expressed by

$$\varepsilon_x = -\kappa y \tag{2.32}$$

where κ, the curvature of the beam axis, is

$$\kappa = \frac{1}{\rho} = \frac{d\theta}{ds} = \frac{w''}{\left[1 + (w')^2\right]^{3/2}} \tag{2.33}$$

with $w(x)$ being the deflection of the beam axis. By Assumption (c), $|w'| \ll 1$ and Eq. (2.33) reduces to

$$\kappa = \frac{1}{\rho} = w'' \tag{2.34}$$

Moment-Curvature Relation According to Assumption (b) and Eqs. (2.32) and (2.34), the normal stress is

$$\sigma_x = -Eyw'' \tag{2.35}$$

The bending moment of the beam at x is

$$M(x) = -\int_A \sigma_x y \, dA = Ew'' \int_A y^2 \, dA = EIw'' \tag{2.36}$$

where A is the beam cross-section area and I is the moment of inertia of the area defined by

$$I = \int_A y^2 \, dA \tag{2.37}$$

The EI is known as *bending stiffness* or *flexural rigidity*. According to Eqs. (2.34) and (2.35), the beam curvature and normal stress are related to bending moment by

$$\kappa = \frac{1}{\rho} = \frac{M}{EI} \tag{2.38}$$

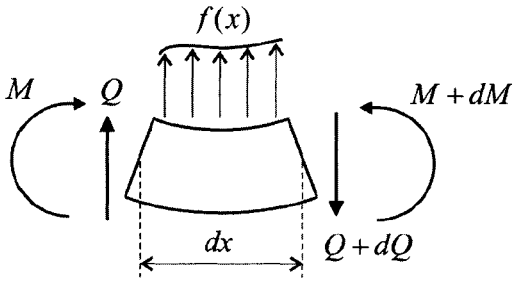


FIGURE 2.2.7 An infinitesimal beam segment.

and

$$\sigma_x = -\frac{M}{I}y \tag{2.39}$$

Equilibrium Equations Consider an infinitesimal beam segment dx in Fig. 2.2.7. Balance of the forces and moments that are applied to the segment gives

$$Q + qdx - (Q + dQ) = 0$$

$$-M - Qdx - qdx \cdot \frac{dx}{2} + M + dM = 0$$

By letting $dx \to 0$, the above equations reduce to

$$\frac{dQ(x)}{dx} = f(x) \tag{2.40}$$

and

$$\frac{dM(x)}{dx} = Q(x) \tag{2.41}$$

It follows from Eqs. (2.36), (2.40), and (2.41) that

$$f(x) = \frac{dQ}{dx} = \frac{d^2M}{dx^2} = EI\frac{d^4w}{dx^4}$$

which is Eq. (2.1), and

$$M = EI\frac{d^2}{dx^2}w, \quad Q = EI\frac{d^3}{dx^3}w$$

which are Eqs. (2.4) and (2.5).

2.3 Static Analysis by the Toolbox

This chapter has a toolbox of MATLAB functions for determination of the static response of an Euler-Bernoulli beam under external loads and boundary disturbances. Solution by the

Toolbox takes two simple steps:

Step 1. Set up system (beam parameters and boundary conditions) by function **setbeam**.
Step 2. Determine beam response by functions **beamf** and **beamb**.

Also, function **plotbeam** can be called to display beam response. These MATLAB functions, along with others from the Toolbox, will be introduced in sequel.

2.3.1 System Setup

In static analysis of a beam, function **setbeam** from the Toolbox is always called first to set up beam parameters and boundary conditions; see Window 3.1.

Window 3.1. Function setbeam

MATLAB Function: **setbeam**

Purpose:
To set up parameters and boundary conditions for a beam.

Synopsis:

```
setbeam(EI, L, BC_Spec)
setbeam(EI, L, BC_Spec, n)
```

Description:
`setbeam(EI, L, BC_Spec)` inputs the bending stiffness `EI` and length `L` of a beam, and specifies boundary conditions by vector `BC_Spec`. The formation of `BC_Spec` is done according to Table 2.3.1.
`setbeam(EI, L, BC_Spec, n)` sets up `n` equally-spaced points along the beam length for graphic purposes. By default, `n = 201`.

Note:
The information of beam parameters and boundary conditions can be displayed by function `systinfo`; see Window 3.9.

The spatial points for computation can be changed by functions **setxpt** or **setxpt1**; see Window 3.10.

Note 1. As an argument of `setbeam`, vector `BC_Spec` is of the form

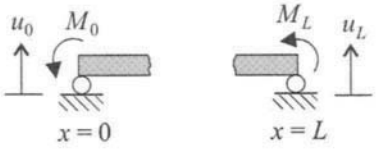
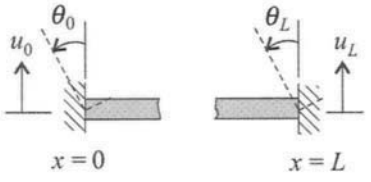
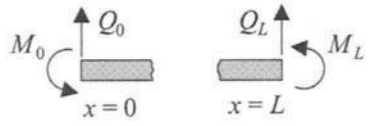
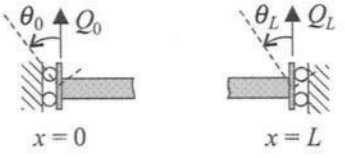
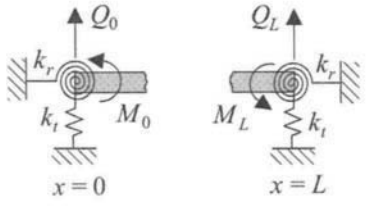
```
BC_Spec = [BC_Left   BC_Right]
```

where `BC_Left` and `BC_Right` are two subvectors describing the boundary conditions at the left end ($x = 0$) and the right end ($x = L$) of the beam, respectively. Table 2.3.1 lists the entries of these vectors for the seven different boundary conditions (B1 to B7). The first element of `BC_Left` and `BC_Right` is an integer identifying the boundary condition type.

Note 2. Function `setbeam` must be executed before any other functions of static analysis from the Toolbox can be used. However, `setbeam` only needs to be called once for the same beam in different cases of external loads or boundary disturbances.

TABLE 2.3.1 Boundary Conditions and Boundary Disturbances

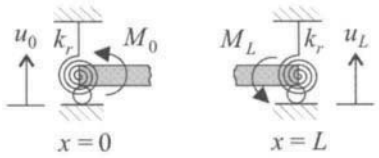
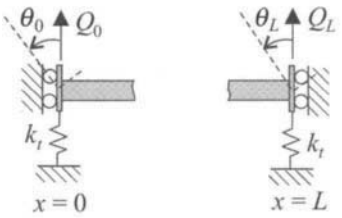
Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`
Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

Boundary Type	*Left Boundary (x = 0)*		*Right Boundary (x = L)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B1) Pinned or hinged end 	[1]	$[u_0 \quad M_0]$	[1]	$[u_L \quad M_L]$
(B2) Clamped or fixed end 	[2]	$[u_0 \quad \theta_0]$	[2]	$[u_L \quad \theta_L]$
(B3) Free end 	[3]	$[M_0 \quad Q_0]$	[3]	$[M_L \quad Q_L]$
(B4) Sliding or guided end 	[4]	$[\theta_0 \quad Q_0]$	[4]	$[\theta_L \quad Q_L]$
(B5) Elastic end 	[5 k_r k_t]	$[M_0 \quad Q_0]$	[5 k_r k_t]	$[M_L \quad Q_L]$

(continued)

TABLE 2.3.1 Boundary Conditions and Boundary Disturbances (*Continued*)

Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`
Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

Boundary Type	*Left Boundary (x = 0)*		*Right Boundary (x = L)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B6) Hinged-elastic end 	$[6\ \ k_r]$	$[u_0\ \ M_0]$	$[6\ \ k_r]$	$[u_L\ \ M_L]$
(B7) Sliding-elastic end 	$[7\ \ k_t]$	$[\theta_0\ \ Q_0]$	$[7\ \ k_t]$	$[\theta_L\ \ Q_L]$

u_0, u_L - end displacement; θ_0, θ_L - end rotation (in radians);
M_0, M_L - end moment; Q_0, Q_L - end transverse force;
k_r - torsional spring coefficient; k_t - translational spring coefficient.
Positive direction of end displacement and transverse force: `upward`.
Positive direction of end rotation and moment: `counterclockwise`.

EXAMPLE 3.1

For the boundary conditions given in Fig. 2.3.1, the boundary specification vector is:

Case (a) Pinned-pinned beam (B1-B1):

```
BC_Spec = [1  1]
```

Case (b) Clamped-free beam (B2-B3):

```
BC_Spec = [2  3]
```

Case (c) Clamped–sliding-elastic (B2-B7):

```
BC_Spec = [1  7  kt]
```

2.3.2 Response to External Forces

This Toolbox provides a function `beamf` for computing the static response of a beam under external forces; see Windows 3.2.

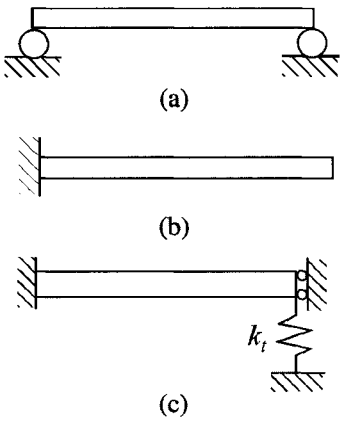


FIGURE 2.3.1 The beam boundary conditions in Example 3.1.

Window 3.2. Function beamf

MATLAB Function: **beamf**

Purpose:
To compute and display static response of a beam under external forces.

Synopsis:

```
beamf(Load_Spec)
y = beamf(Load_Spec)
[y, x] = beamf(Load_Spec)
```

Description:
`beamf(Load_Spec)` plots the spatial distribution of beam response subject to an external load specified by vector `Load_Spec`. See Table 2.3.2 for the formation of `Load_Spec`.

`y = beamf(Load_Spec)` returns the computed beam response in a matrix `y`. The beam response distribution can be plotted by `plotbeam(y)`; see Window 3.3.

`[y, x] = beamf(Load_Spec)` returns the beam response in a four-row matrix `y`, and the spatial points of computation in a vector `x`. Here, `y(1,:)` stores beam displacement, `y(2,:)` rotation, `y(3,:)` bending moment, and `y(4,:)` shear force. With `y` and `x`, the spatial distribution of the beam response can be plotted by the MATLAB function `plot`. For example, for the deflection $w(x)$, use `plot(x, y(1,:))`; and for the shear force $Q(x)$, use `plot(x, y(4,:))`.

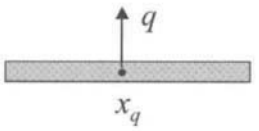
If a beam is subject to N external loads specified by vectors `Load1`, `Load2`,..., `LoadN`, its response can be obtained by superposition:

```
y = beamf(Load1) + beamf(Load2) + ··· + beamf(LoadN)
```

and the spatial distribution of the beam response is plotted by function `plotbeam`. See Section 2.3.4 for more information.

TABLE 2.3.2 Specification of External Loads

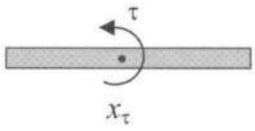
(L1) Pointwise force



$f(x) = q\delta(x - x_q)$

Load_Spec $= [0 \quad x_q \quad q]$

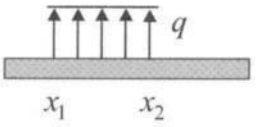
(L2) Torque



$f(x) = -\tau \dfrac{d}{dx}\delta(x - x_\tau)$

Load_Spec $= [-1 \quad x_\tau \quad \tau]$

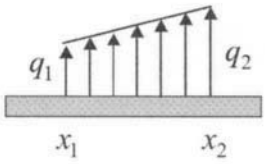
(D1) Uniform



$f(x) = q \quad \text{for } x_1 \le x \le x_2$

Load_Spec $= [1 \quad x_1 \quad x_2 \quad q]$

(D2) Linear



$f(x) = q_1 + \dfrac{q_2 - q_1}{x_2 - x_1}(x - x_1) \quad \text{for } x_1 \le x \le x_2$

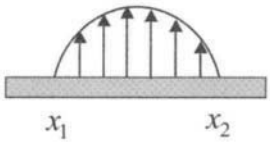
Load_Spec $= [2 \quad x_1 \quad x_2 \quad q_1 \quad q_2]$

(D3) Quadratic

$f(x) = c_0 + c_1 x + c_2 x^2 \quad \text{for } x_1 \le x \le x_2$

Load_Spec $= [3 \quad x_1 \quad x_2 \quad c_0 \quad c_1 \quad c_2]$

(D4) Sinusoidal



$f(x) = A\sin(\omega x + \phi) \quad \text{for } x_1 \le x \le x_2$

Load_Spec $= [4 \quad x_1 \quad x_2 \quad A \quad \omega \quad \phi]$

(ϕ in radians)

As an argument of `beamf`, vector `Load_Spec` describes one of the six types of external forces listed in Table 2.3.2. Note that the first element of `Load_Spec` is an integer from -1 to 4 for identifying load type. In the table, there are two pointwise loads (L1 and L2) and four distributed loads (D1 to D4). The sign convention of external forces is as follows:

- The positive direction of transverse loads (L1, D1 to D4) is upward.
- The positive direction of a torque (L2) is counterclockwise.

EXAMPLE 3.2

For the load of parabolic distribution shown in Fig. 2.3.2,

$$f(x) = q\frac{(x - x_1)^2}{(x_2 - x_1)^2}, \quad x_1 \le x \le x_2$$

the load specification vector

$$\text{Load\_Spec} = [3 \quad x_1 \quad x_2 \quad c_0 \quad c_1 \quad c_2]$$

where $c_0 = \frac{q}{a^2}x_1^2$, $c_1 = -\frac{2q}{a^2}x_1$, $c_2 = \frac{q}{a^2}$, $a = x_2 - x_1$.

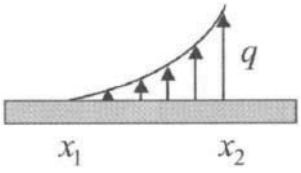


FIGURE 2.3.2 A load of parabolic distribution.

The beam response obtained by `beamf` can be plotted by function `plotbeam`, as described in Window 3.3.

Window 3.3. Function `plotbeam`

MATLAB Function: **`plotbeam`**

Purpose:
To plot spatial distribution of beam response and to show maximum beam response and reactions at beam supports.

Synopsis:

```
plotbeam(y)
plotbeam(y, opt)
```

Description:
`plotbeam(y)` carries out the following three tasks: (a) it plots spatial distributions of beam response; (b) it determines maximum beam response; and (c) it gives the reactions at beam supports. Here `y` is a matrix of beam response data that is obtained through use of `beamf` or `beamb`. The output of `plotbeam(y)` is four figures, namely the displacement, rotation, bending moment, and shear force versus the spatial coordinate x.

`plotbeam(y, opt)` plots beam response with the following options:

`opt` = 1	plot displacement $w(x)$ only
`opt` = 2	plot rotation $\theta(x)$ only
`opt` = 3	plot bending moment $M(x)$ only
`opt` = 4	plot shear force $Q(x)$ only

EXAMPLE 3.3

In Fig. 2.3.3, a beam is clamped at the left end and elastically supported at the right end by a spring of coefficient k_t = 375. The beam is subject to a force of half-sine

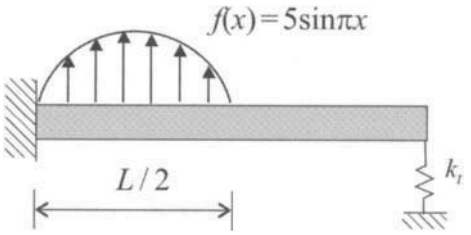


FIGURE 2.3.3 A beam subject to a force of half-sine distribution.

distribution:

$$f(x) = 5\sin(\pi x), \quad 0 \le x \le 1$$

Let the beam parameters be $EI = 800$ and $L = 2$. By the commands (typed in the MATLAB command window)

```
>> setbeam(800, 2, [2 5 0 375])
>> beamf([4  0  1  5  pi  0])
```

the beam response versus x is plotted in Fig. 2.3.4. Also,

```
>> y = beamf([4  0  1  5  pi  0]);
>> plotbeam(y)
```

yields Fig. 2.3.4, and gives the following information about the computed results:

```
Maximum Beam Response in Absolute Value
---------------------------------------
Max displacement = 0.00046807,  location x = 2
Max rotation (degrees) = 0.020213,  location x = 0.52
Max bending moment = 1.2405,  location x = 0
Max shear force = -3.0076,  location x = 0
```

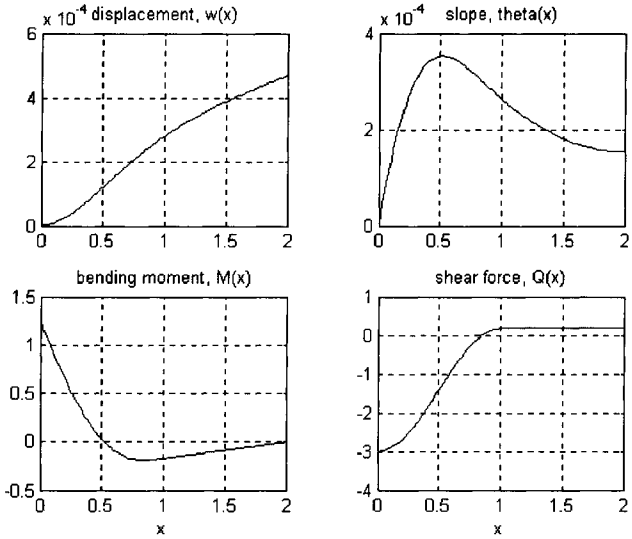


FIGURE 2.3.4 Spatial distribution of beam response.

```
Reactions at Two Ends of the Beam
--------------------------------
Sign convention for support reactions:
* Positive moment Mc: counterclockwise
* Positive force Rc: upward
At left boundary (x = 0):   Mc = -1.2405,  Rc = -3.0076
At right boundary (x = L):  Mc = 0,  Rc = -0.17552
```

2.3.3 Response to Boundary Disturbances

To compute the static response of a beam under boundary disturbances, use function `beamb` from the Toolbox; see Windows 3.4. The boundary disturbances are specified by vector

```
BD_Spec = [BD_Left BD_Right]
```

where the subvectors `BD_Left` and `BD_Right` describe the disturbances at the left end ($x = 0$) and right end ($x = L$) of the beam, respectively; see Table 2.3.1. The end rotations θ_0 and θ_L in the table are in radians.

Window 3.4. Function beamb

MATLAB Function: **beamb**

Purpose:
To compute and display static response of a beam under boundary disturbances.

Synopsis:

```
beamb(BD_Spec)
y = beamb(BD_Spec)
[y, x] = beamb(BD_Spec)
```

Description:
`beamb(BD_Spec)` plots spatial distribution of the response of a beam subject to boundary disturbances, where `BD_Spec` is a vector specifying given boundary disturbances according to Table 2.3.1.

`y = beamb(BD_Spec)` returns the computed beam response in a matrix `y`. The beam response distribution can be plotted by `plotbeam(y)`, which is described in Window 3.3.

`[y, x] = beamb(BD_Spec)` returns the beam response in a four-row matrix `y`, and the points of computation in a vector `x`. Here, `y(1,:)` stores beam displacement, `y(2,:)` rotation, `y(3,:)` bending moment, and `y(4,:)` shear force. With `y` and `x`, the spatial distribution of beam response can be plotted by the MATLAB function `plot`. For example, for deflection $w(x)$, use `plot(x, y(1, :))`; and for bending moment $M(x)$, use `plot(x, y(3, :))`.

EXAMPLE 3.4

In Fig. 2.3.5, a clamped-pinned beam of bending stiffness $EI = 40$ and length $L = 4$ is subject to the boundary disturbances

$$\theta_0 = 0.1, \quad u_L = 0.05, \quad M_L = 15.$$

FIGURE 2.3.5 A clamped-pinned beam subject to boundary disturbances.

The boundary disturbance vector, according to Table 2.3.1, is

```
BD_Spec = [0 0.1 0.05 15]
```

```
>> setbeam(40, 4, [2 1])
>> beamb([0 0.1 0.05 15])
```

yields the spatial distribution of the beam response plotted in Fig. 2.3.6.

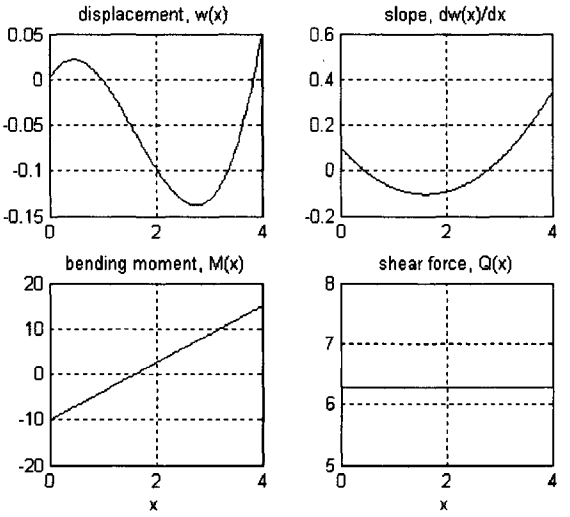


FIGURE 2.3.6 Spatial distribution of beam response.

2.3.4 Total Response

When a beam is subject to an external load and boundary disturbances, the total response of the beam can be obtained and plotted by

```
y = beamf(Load_Spec)+beamb(BD_Spec);
plotbeam(y, opt)
```

where vectors `Load_Spec` and `BD_Spec` are formed according to Tables 2.3.1 and 2.3.2, respectively, and function `plotbeam` is given in Window 3.3.

If a beam is subject to N external loads specified by `Load1, Load2,..., LoadN` and boundary disturbances specified by `BD_Spec`, the total response of the beam is obtained by

```
y = beamf(Load1) + beamf(Load2) + ··· + beamf(LoadN) + beamb(BD_Spec);
```

Here, y is a four-row matrix, with y(1,:) containing the beam displacement, y(2,:) the rotation, y(3,:) the bending moment, and y(4,:) the shear force. There are three ways to plot the spatial distribution of the beam response:

(i) Use function plotbeam(y, opt).
(ii) Use the MATLAB function plot as follows:

```
[yb, x] = beamb(BD_Spec);
y = beamf(Load1) + ··· + beamf(LoadN) + yb;
plot(x, y(j,:))
```

where j is an integer from 1 to 4.
(iii) Use function getxpt from the Toolbox:

```
y = beamf(Load1) + ··· + beamf(LoadN) + beamb(BD_Spec);
x = getxpt;
plot(x, y(j,:))
```

where j is an integer from 1 to 4 and getxpt is a function described in Window 3.11.

EXAMPLE 3.5

In Fig. 2.3.7, a beam of parameters $EI = 800$ and $L = 2$ is clamped at its left end and elastically supported at its right end by a translational spring of coefficient $k_t = 375$. There are two external loads: a torque of $\tau_0 = 1.5$ located at $x_\tau = 1.0$ and a uniform load of amplitude $q = 1.8$ applied in the region $1.0 \leq x \leq 2.0$. The beam is also subject to a rotation at its left end, $\theta_0 = 0.01$ rad. By

```
>> setbeam(800, 2, [2 5 0 375]);
>> Load1 = [-1 1.0 1.5]; Load2 = [1 1 2 1.8]; BD_Spec = [0 0.01 0 0];
>> y = beamf(Load1) + beamf(Load2) + beamb(BD_Spec);
>> plotbeam(y)
```

the total beam response is plotted in Fig. 2.3.8. In addition, the following results are obtained:

```
Maximum Beam Response in Absolute Value
-----------------------------------------------------------
Max displacement = 0.011847,  location x = 2
Max rotation (degrees) = 0.57296,  location x = 0
Max bending moment = -4.6854,  location x = 0
Max shear force = 4.4427,  location x = 2
```

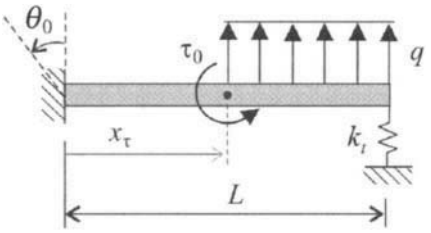


FIGURE 2.3.7 A beam under a uniform load q, a torque τ_0 and boundary disturbance θ_0.

```
Reactions at Two Ends of the Beam
-----------------------------------------------
Sign convention for support reactions:
* Positive moment Mc: counterclockwise
* Positive force Rc: upward
At left boundary (x = 0):
      Mc = 4.6854,  Rc = 2.6427
At right boundary (x = L):
      Mc = 0,  Rc = -4.4427
```

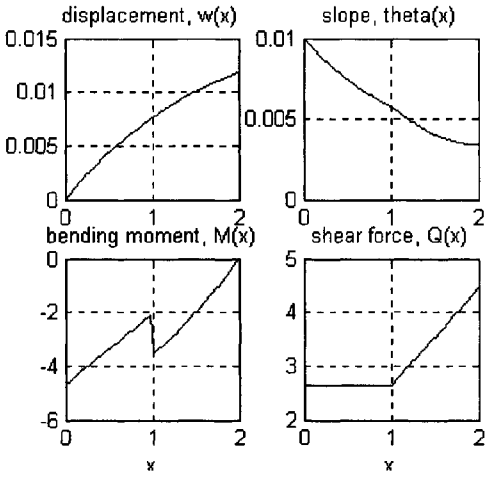


FIGURE 2.3.8 Total beam response.

2.3.5 Exact Analytical Solutions

Functions `beamf` and `beamb` obtain beam response numerically. For exact analytical expressions of beam response under arbitrary boundary conditions and external loads, two other functions from the Toolbox are available; see Windows 3.5 and 3.6.

Window 3.5. Function `mathf`

MATLAB Function: **`mathf`**

Purpose:
To display mathematical expressions of beam response to an external force.

Synopsis:

```
mathf(Load_Spec)
y = mathf(Load_Spec)
```

Description:
`mathf(Load_Spec)` displays the mathematical expressions of the response (displacement, rotation, bending moment, and shear force) of a beam subject to an external load specified by vector `Load_Spec`. Formation of `Load_Spec` follows Table 2.3.2.

`y = mathf(Load_Spec)` returns the coefficients in the mathematical expressions of the beam response in a matrix `y`.

Window 3.6. Function mathb

MATLAB Function: **mathb**

Purpose:
To display mathematical expressions of beam response to boundary disturbances.

Synopsis:

```
mathb(BD_Spec)
y = mathb(BD_Spec)
```

Description:
`mathb(BD_Spec)` displays the mathematical expressions of the response (displacement, rotation, bending moment, and shear force) of a beam subject to boundary disturbances specified by vector `BD_Spec`. Formation of `BD_Spec` follows Table 2.3.1.

`y = mathb(BD_Spec)` returns the coefficients in the mathematical expressions of the beam response in a matrix `y`.

EXAMPLE 3.6

Consider the beam in Fig. 2.1.1, Section 2.1. By

```
>> setbeam(25, 1, [1 1])
>> mathf([0 0.5 1.2])
```

the mathematical expressions of the beam response are obtained as follows:

```
Beam response for 0  <= x <= 0.5:
 Displacement    w(x) = -0.004*x^3 + (0)*x^2 + (0.003)*x + (0)
 Rotation    dw(x)/dx = -0.012*x^2 + (0)*x + (0.003)
 Bending moment  M(x) = -0.6*x + (0)
 Shear force     Q(x) = -0.6
Beam response for 0.5 <= x <= 1:
 Displacement    w(x) = 0.004*x^3 + (-0.012)*x^2 + (0.009)*x + (-0.001)
 Rotation    dw(x)/dx = 0.012*x^2 + (-0.024)*x + (0.009)
 Bending moment  M(x) = 0.6*x + (-0.6)
 Shear force     Q(x) = 0.6
```

EXAMPLE 3.7

For the beam in Example 3.3

```
>> setbeam(800, 2, [2 5 0 375])
>> mathf([4  0  1  5  pi  0])
```

gives the beam response as follows:

```
Beam response for 0 <= x <= 1:
 Displacement   w(x) = 6.4162e-005*sin(3.1416*x + 0) + (-0.00029501)*x^3
                      + (0.00077531)*x^2 + (-0.00020157)*x + (0)
 Rotation   dw(x)/dx = 0.00020157*cos(3.1416*x+0) + (-0.00088502)*x^2
                      + (0.0015506)*x + (-0.00020157)
 Bending moment M(x) = -0.50661*sin(3.1416*x+0) + (-1.416)*x + (1.2405)
 Shear force    Q(x) = -1.5915*cos(3.1416*x+0) + (-1.416)
Beam response for 1 <= x <= 2:
 Displacement w(x)   = 3.6568e-005*x^3 + (-0.00021941)*x^2
                      + (0.00059157)*x + (-0.00013)
 Rotation   dw(x)/dx = 0.0001097*x^2 + (-0.00043881)*x + (0.00059157)
 Bending moment M(x) = 0.17552*x + (-0.35105)
 Shear force    Q(x) = 0.17552
```

EXAMPLE 3.8

In Fig. 2.3.9, a clamped-pinned beam of parameters $EI = 250$ and $a = 1$ is subject to a linearly distributed load ($q = 10$) in the region $a \leq x \leq 2a$. By

```
>> setbeam(250, 3, [2 1])
>> mathf([2 1 2 -10 0])
```

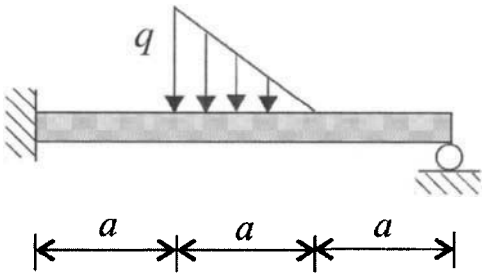


FIGURE 2.3.9 A clamped-pinned beam subject to a linearly distributed load.

the mathematical expressions of the beam response are obtained as follows:

```
Beam response for 0 <= x <= 1:
 Displacement   w(x) = 0.0024753*x^3 + (-0.0056111)*x^2 + (0)*x + (0)
 Rotation   dw(x)/dx = 0.0074259*x^2 + (-0.011222)*x + (0)
 Bending moment M(x) = 3.713*x + (-2.8056)
 Shear force    Q(x) = 3.713
Beam response for 1 <= x <= 2:
 Displacement   w(x) = 0.00033333*x^5 + (-0.0033333)*x^4 + (0.012475)*x^3
                      + (-0.018944)*x^2 + (0.0083333)*x + (-0.002)
 Rotation   dw(x)/dx = 0.0016667*x^4 + (-0.013333)*x^3 + (0.037426)*x^2
                      + (-0.037889)*x + (0.0083333)
 Bending moment M(x) = 1.6667*x^3 + (-10)*x^2 + (18.713)*x + (-9.4722)
 Shear force    Q(x) = 5*x^2 + (-20)*x + (18.713)
```

```
Beam response for 2 <= x <= 3:
 Displacement   w(x) = -0.00085802*x^3 + (0.0077222)*x^2 + (-0.018333)*x
                       + (0.0086667)
 Rotation    dw(x)/dx = -0.0025741*x^2 + (0.015444)*x + (-0.018333)
 Bending moment M(x) = -1.287*x + (3.8611)
 Shear force    Q(x) = -1.287
```

For comparison, the beam response is plotted in Fig. 2.3.10 by

```
>> beamf([2 1 2 -10 0])
```

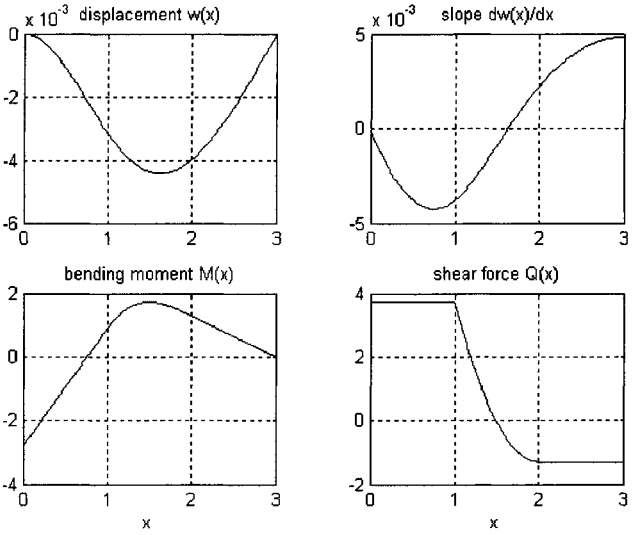


FIGURE 2.3.10 Beam response to a linearly distributed load.

EXAMPLE 3.9

Consider the same beam and boundary disturbances given in Example 3.4. By

```
>> setbeam(40, 4, [2 1])
>> mathb([0 0.1 0.05 15])
```

the mathematical expressions of the beam response is obtained as follows:

```
Beam response for 0 <= x <= 4:
 Displacement    w(x) = 0.026172*x^3 + (-0.12656)*x^2 + (0.1)*x + (0)
 Rotation     dw(x)/dx = 0.078516*x^2 + (-0.25312)*x + (0.1)
 Bending moment  M(x) = 6.2813*x + (-10.125)
 Shear force     Q(x) = 6.2813
```

2.3.6 Other Useful Functions

This Toolbox has functions `maxresp` and `reactf` for computing maximum response and reaction forces of a beam; see Windows 3.7 and 3.8. These functions are automatically called when function `plotbeam` is used.

Window 3.7. Function `maxresp`

MATLAB Function: **`maxresp`**

Purpose:
To determine maximum beam response in absolute value and corresponding location.

Synopsis:

```
maxresp(y)
```

Description:
`maxresp(y)` displays (in the MATLAB window) maximum beam response (displacement, rotation, bending moment, and shear force) in absolute value and the corresponding locations. Here `y` is a matrix of beam response data that are obtained by `beamf` and/or `beamb`.

Window 3.8. Function `reactf`

MATLAB Function: **`reactf`**

Purpose:
To determine reaction forces at beam boundaries.

Synopsis:

```
reactf(y)
```

Description:
`reactf(y)` displays (in the MATLAB window) the reaction forces and displacements of a beam at its two ends. Here `y` is a matrix of beam response data obtained by `beamf` and/or `beamb`.

In static analysis of a beam, the information of beam parameters and boundary conditions that are set up by function `setbeam` can be retrieved and displayed at any time by function `systinfo`; see Window 3.9.

Window 3.9. Function `systinfo`

MATLAB Function: **`systinfo`**

Purpose:
To display beam parameters, boundary conditions, and number of spatial points selected for computation.

Window 3.9. Function systinfo (*Continued*)

Synopsis:

```
systinfo
```

Functions setxpt and setxpt1 described in Window 3.10 can be called to reset or change the spatial points along beam length, for computation and display purposes. This is done without the need to execute function setbeam again. In addition, function getxpt described in Window 3.11 allows the user to get access to the spatial points set up by function setbeam, setxpt, and setxpt1, which is useful in plotting beam response.

Window 3.10. Functions setxpt and setxpt1

MATLAB Function: **setxpt, setxpt1**

Purpose:
To reset or change spatial points along beam length for computation and display.

Synopsis:

```
setxpt(n)
setxpt1(X_Point)
```

Description:
setxpt(n) sets up n points equally spaced along the length of a beam for computation; namely, $x_k = k \times L/(n-1)$, $k = 0, 1, \ldots, n-1$, where L is the beam length.

setxpt(X_Point) sets up points defined by a vector X_Point, at which beam response is computed and plotted. Those points do not have to be equally spaced. For example,

```
>> X_Point = [0.5 0.6 0.76];
>> setxpt1(X_Point)
```

selects the points at $x = 0.5, 0.6$, and 0.76 for computation and display of beam response.

Window 3.11. Function getxpt

MATLAB Function: **getxpt**

Purpose:
To obtain the spatial points that are set up for computation.

Synopsis:

```
x = getxpt
```

Description:
x = getxpt obtains the spatial points that are set up for computation by one of the functions setbeam, setxpt, or setxpt1. Here x is a vector containing the points.

2.4 Moments of Inertia of Beam Cross-Section Area

Consider a beam cross section in Fig. 2.4.1. The moment of inertia of the area about the centroidal axis z-z is defined as

$$I_z = \int_A y^2 \, dA \tag{2.42}$$

where A is the area of the cross section. The location of the centroidal axis can be determined in any coordinate system, say yoz_1, by

$$y_c = \frac{1}{A} \int_A y \, dA \tag{2.43}$$

The moment of inertia of an area about the axis z_1 is obtained according to the parallel axis theorem

$$I_{z_1} = \int_A (y + y_c)^2 dA = I_z + y_c^2 A \tag{2.44}$$

The Toolbox of this chapter has a function `minertia` for computing the moment of inertia of 10 types of areas; see Window 4.1.

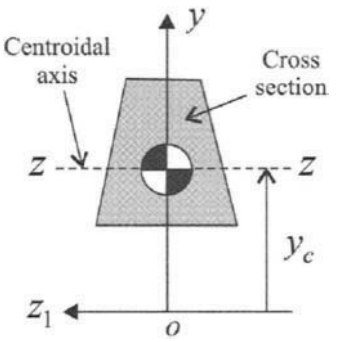


FIGURE 2.4.1 A beam cross section.

Window 4.1. Function `minertia`

MATLAB Function: **`minertia`**

Purpose:
To compute moments of inertia of a given area.

Synopsis:

```
minertia(A_Spec)
[Iz, A, Yc] = minertia(A_Spec)
```

Description:
`minertia(A_Spec)` computes the moment of inertia of an area, where `A_Spec` is a vector specifying the area by

```
A_Spec = [type_no par1 par2 ... .]
```

where `type_no` is an integer (from 1 to 10) identifying the area type, and `par1`, `par2`, ... are geometric parameters of the area; see Table 4.1, where the centroidal axis of an area is represented by a horizontal line z-z.

`[Iz, A, Yc] = minertia(A_Spec)` returns the moment of inertia `Iz`, the total area `A`, and the centroid `Yc` of the area. Here `Yc` is measured from the bottom of the area.

In what follows, the derivation of the moments of inertia for Areas 7 to 9 listed in Table 2.4.1 is provided.

Area 7: T-section (Fig. 2.4.2)

The centroid is at

$$y_c = \frac{A_1(h - t_1/2) + A_2(h - t_1)/2}{A}$$

where A_1 and A_2 are the subareas in Fig. 2.4.2, i.e.,

$$A_1 = bt_1, \quad A_2 = t_2(h - t_1), \quad A = A_1 + A_2$$

The moment of inertia of the area is

$$I_7 = \frac{1}{12}bt_1^3 + \rho_1^2 A_1 + \frac{1}{12}t_2(h - t_1)^3 + \rho_2^2 A_2 \tag{2.45}$$

with $\rho_1 = h - t_1/2 - y_c$, $\rho_2 = |y_c - (h - t_1)/2|$.

Area 8: U-Channel (Fig. 2.4.3)

The centroid is at

$$y_c = \frac{A_1(h - t_1/2) + A_2(h - t_1)}{A}$$

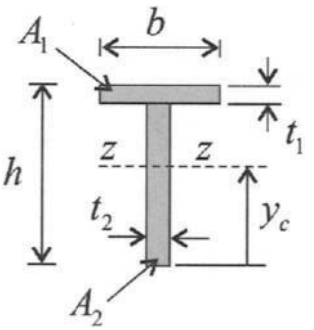
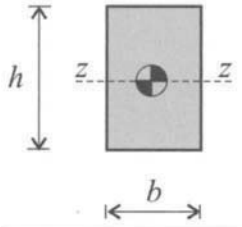


FIGURE 2.4.2 Area 7.

TABLE 2.4.1 Moments of Inertia of Areas Around Centroidal Axes

Area 1: Rectangle

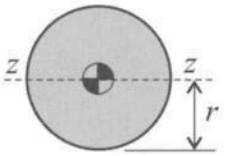


$$A = bh$$

$$I_z = \frac{1}{12}bh^3$$

A_Spec = [1 b h]

Area 2: Circle

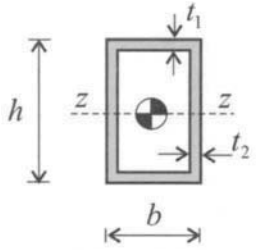


$$A = \pi r^2$$

$$I_z = \frac{1}{4}\pi r^4$$

A_Spec = [2 r]

Area 3: Hollow rectangle



$$A = 2t(b + h - 2t)$$

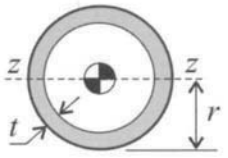
$$I_z = \frac{1}{12}\left(bh^3 - b_1h_1^3\right)$$

where

$$b_1 = b - 2t_2,\ h_1 = h - 2t_1$$

A_Spec = [3 b h t1 t2]

Area 4: Thin tube

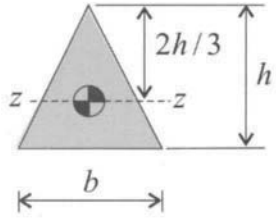


$$A = \pi(2r - t)t$$

$$I_z = \frac{\pi}{4}\left(r^4 - (r - t)^4\right)$$

A_Spec = [4 r t]

Area 5: Triangle

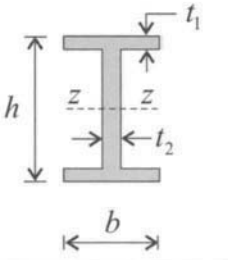


$$A = \frac{1}{2}bh$$

$$I_z = \frac{1}{36}bh^3$$

A_Spec = [5 b h]

Area 6: I-shape



$$A = 2bt_1 + t_2(h - 2t_1)$$

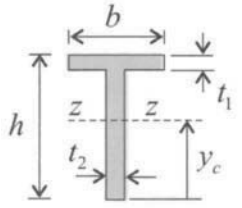
$$I_z = \frac{1}{12}\left(bh^3 - b_1h_1^3\right)$$

where

$$b_1 = b - t_2,\ h_1 = h - 2t_1$$

A_Spec = [6 b h t1 t2]

Area 7: T-section

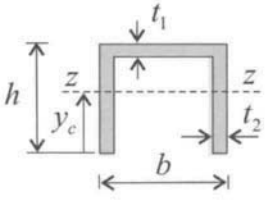


$$A = bt_1 + t_2(h - t_1)$$

$$I_z = I_7 \text{ given in Eq. (2.45)}$$

A_Spec = [7 b h t1 t2]

Area 8: U-channel

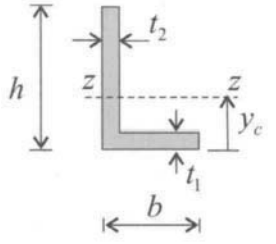


$$A = 2t(b + h - 2t)$$

$$I_z = I_8 \text{ given in Eq. (2.46)}$$

A_Spec = [8 b h t1 t2]

Area 9: L-shape

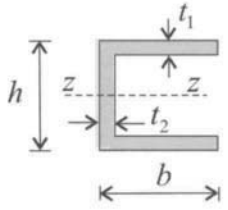


$$A = ht_2 + (b - t_2)t_1$$

$$I_z = I_9 \text{ given in Eq. (2.47)}$$

A_Spec = [9 b h t1 t2]

Area 10: C-shape



$$A = 2bt_1 + (h - 2t_1)t_2$$

$$I_z = \frac{1}{12}\left(bh^3 - b_1h_1^3\right)$$

where

$$b_1 = b - t_2,\ h_1 = h - 2t_1$$

A_Spec = [10 b h t1 t2]

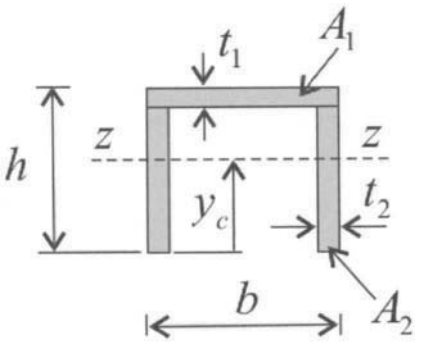


FIGURE 2.4.3 Area 8.

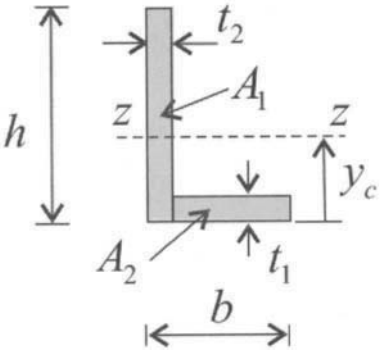


FIGURE 2.4.4 Area 9.

where A_1 and A_2 are the subareas in Fig. 2.4.3, i.e.,

$$A_1 = bt_1, \quad A_2 = t_2(h - t_1), \quad A = A_1 + 2A_2$$

The moment of inertia of the area is

$$I_7 = \frac{1}{12}bt_1^3 + \rho_1^2 A_1 + 2\left(\frac{1}{12}t_2(h - t_1)^3 + \rho_2^2 A_2\right) \tag{2.46}$$

with $\rho_1 = h - t_1/2 - y_c, \quad \rho_2 = |y_c - (h - t_1)/2|$.

Area 9: L-shape (Fig. 2.4.4)

The centroid is at

$$y_c = \frac{A_1 h/2 + A_2 t_1/2}{A}$$

where A_1 and A_2 are the subareas in Fig. 2.4.4, i.e.,

$$A_1 = ht_2, \quad A_2 = (b - t_2)t_1, \quad A = A_1 + A_2$$

The moment of inertia of the area is

$$I_9 = \frac{1}{12}t_2h^3 + \frac{1}{12}(b - t_2)t_1^3 + \rho_1^2A_1 + \rho_2^2A_2 \tag{2.47}$$

with $\rho_1 = |y_c - h/2|$, $\rho_2 = |y_c - t_1/2|$.

EXAMPLE 4.1

For the 10 areas in Table 2.4.1, assume

$$b = 0.2, \quad h = 0.5, \quad r = 0.3, \quad t = t_1 = t_2 = 0.05.$$

The moment of inertia, area, and centroid computed by `minertia` are given in Table 2.4.2.

TABLE 2.4.2

Cross Section	*Moment of Inertia* I_z	*Area A*	*Centroid* y_c
Area 1: Rectangle	0.0020833	0.1	0.25
Area 2: Circle	0.0063617	0.28274	0.15
Area 3: Hollow rectangle	0.00155	0.06	0.25
Area 4: Thin tube	0.0032938	0.086394	0.15
Area 5: Triangle	0.00069444	0.05	0.16667
Area 6: I-shape	0.0012833	0.04	0.25
Area 7: T-section	0.00081446	0.0325	0.30192
Area 8: U-channel	0.0012728	0.055	0.27045
Area 9: L-shape	0.00081446	0.0325	0.19808
Area 10: C-shape	0.0012833	0.003475	0.25

2.5 Quick Solution Guide

System Description
Equation of motion:

$$EI\frac{d^4}{dx^4}w(x) = f(x), \quad 0 \le x \le L$$

Boundary conditions:

$$\begin{array}{lll} \text{at the left end} & B_{01}\,[w(x)]_{x=0} = \alpha_1, & B_{02}\,[w(x)]_{x=0} = \alpha_2 \\ \text{at the right end} & B_{L1}\,[w(x)]_{x=L} = \beta_1, & B_{L2}\,[w(x)]_{x=L} = \beta_2 \end{array}$$

where B_{01}, B_{02}, B_{L1}, and B_{L2} are spatial differential operators, and α_i and β_i are boundary disturbances.

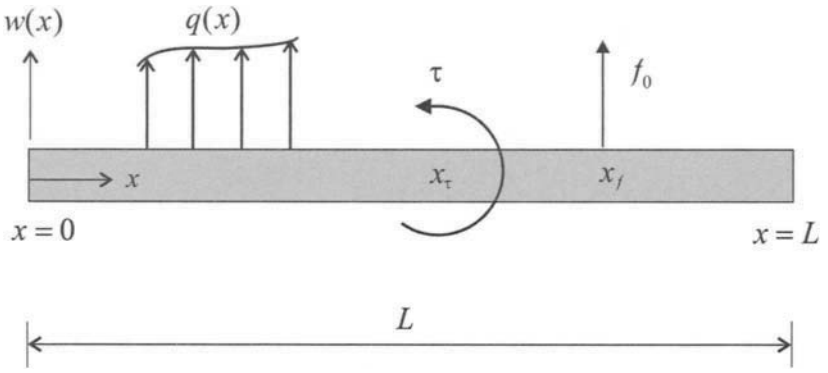


FIGURE 2.5.1 Schematic of Euler-Bernoulli beam.

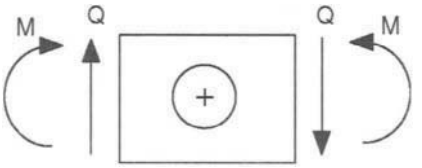


FIGURE 2.5.2 Sign convention of bending moment and shear force.

Problem to Solve

Determination of static response (displacement, rotation, bending moment, and shear force)

Displacement (transverse deflection): $w(x)$

Rotation: $\theta(x) = \dfrac{d}{dx}w(x)$

Bending moment: $M(x) = EI\dfrac{d^2}{dx^2}w(x)$

Shear force: $Q(x) = EI\dfrac{d^3}{dx^3}w(x)$

MATLAB Functions

This chapter contains a Toolbox of MATLAB functions for static analysis of Euler-Bernoulli beams; see the table below. Refer to the corresponding window or section for the utility of each function. The application of some major functions is summarized in the following pages.

	System Setup	
`setbeam`	Set up beam parameters and boundary conditions	Window 3.1
	Static Analysis of Beams	
`beamf`	Compute beam response to external loads	Window 3.2
`beamb`	Compute beam response to boundary disturbances	Window 3.4
`mathf`	Obtain exact analytical expressions of beam response subject to an external load	Window 3.5
`mathb`	Obtain exact analytical expressions of beam response subject to boundary disturbances	Window 3.6
`maxresp`	Determine maximum beam response and location	Window 3.7
`reactf`	Obtain reaction forces at beam boundaries	Window 3.8
`plotbeam`	Plot spatial distribution of beam response	Window 3.3
`minertia`	Compute moments of inertia of beam cross section area	Window 4.1

Utilities		
systinfo	Display beam parameters and boundary conditions	Window 3.9
setxpt setxpt1	Select or reset spatial points for computation	Window 3.10
getxpt	Get access to spatial points that are set up for computation	Window 3.11
TBdemo	Show how the Toolbox works and what it can do	Section 1
TBinfo	Display the information of the Toolbox	Section 1
RunEx	Run all the numerical examples contained in this chapter	Section 1

Solution Procedure

Computation and display of the static response of a beam take the following two steps:

Step 1. Set up system parameters and boundary conditions.

```
>> setbeam(EI, L, BC_Spec)
```

Step 2. Compute and plot beam response.
To compute beam response subject to an external force, type

```
>> beamf(Load_Spec)
```

To compute beam response subject to boundary disturbances, type

```
>> beamb(BD_Spec)
```

To obtain analytical expressions of beam response to an external force, type

```
>> mathf(Load_Spec)
```

For total response to multiple external forces and boundary disturbances, type

```
>> y = beamf(Load1) + ··· + beamf(LoadN) + beamb(BD_Spec);
>> plotbeam(y)
```

Note 1. `setbeam`, `beamf`, `beamb`, and `plotbeam` are functions from the Toolbox.

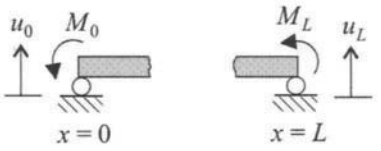
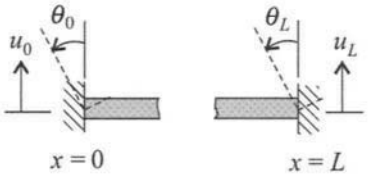
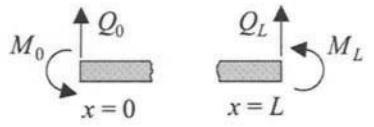
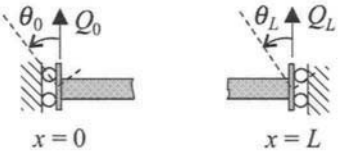
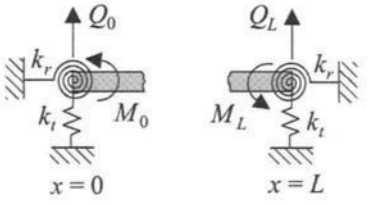
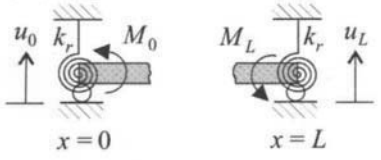
Note 2. `EI` and `L` are the bending stiffness and length of the beam; `BC_Spec` is a vector specifying the beam boundary conditions; `Load_Spec`, `Load1`, `Load2`,..., `LoadN` are vectors specifying external loads; and `BD_Spec` is a vector specifying boundary disturbances. The specification of these vectors is described in Tables 2.5.1 and 2.5.2.

Sign convention for external and boundary disturbances: Positive direction of displacement and transverse force: **upward** ↑

Positive direction of rotation, moment, and torque: **counterclockwise** ↺

Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`

Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

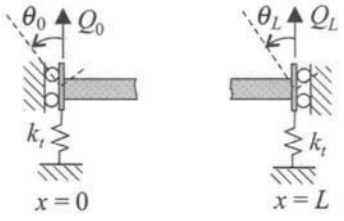
Boundary Type	*Left Boundary (x = 0)*		*Right Boundary (x = L)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B1) Pinned or hinged end 	[1]	$[u_0 \quad M_0]$	[1]	$[u_L \quad M_L]$
(B2) Clamped or fixed end 	[2]	$[u_0 \quad \theta_0]$	[2]	$[u_L \quad \theta_L]$
(B3) Free end 	[3]	$[M_0 \quad Q_0]$	[3]	$[M_L \quad Q_L]$
(B4) Sliding or guided end 	[4]	$[\theta_0 \quad Q_0]$	[4]	$[\theta_L \quad Q_L]$
(B5) Elastic end 	[5 k_r k_t]	$[M_0 \quad Q_0]$	[5 k_r k_t]	$[M_L \quad Q_L]$
(B6) Hinged-elastic end 	[6 k_r]	$[u_0 \quad M_0]$	[6 k_r]	$[u_L \quad M_L]$

(continued)

Specification of Boundary Conditions and Boundary Disturbances (*Continued*)

Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`

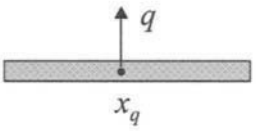
Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

Boundary Type	*Left Boundary (x = 0)*		*Right Boundary (x = L)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B7) Sliding-elastic end 	$[7 \quad k_t]$	$[\theta_0 \quad Q_0]$	$[7 \quad k_t]$	$[\theta_L \quad Q_L]$

u_0, u_L - end displacement; θ_0, θ_L - end rotation (in radians);
M_0, M_L - end moment; Q_0, Q_L - end transverse force;
k_r - torsional spring coefficient; k_t - translational spring coefficient.
Positive direction of end displacement and transverse force: `upward`.
Positive direction of end rotation and moment: `counterclockwise`.

Specification of External Loads

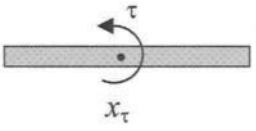
(L1) Pointwise



$f(x) = q\delta(x - x_q)$

Load_Spec = $[0 \quad x_q \quad q]$

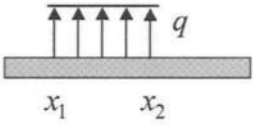
(L2) Torque



$f(x) = -\tau \dfrac{d}{dx}\delta(x - x_\tau)$

Load_Spec = $[-1 \quad x_\tau \quad \tau]$

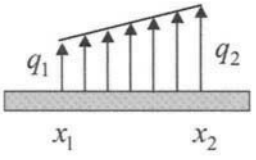
(D1) Uniform



$f(x) = q \quad \text{for } x_1 \le x \le x_2$

Load_Spec = $[1 \quad x_1 \quad x_2 \quad q]$

(D2) Linear



$f(x) = q_1 + \dfrac{q_2 - q_1}{a}(x - x_1)$ for $x_1 \le x \le x_2$

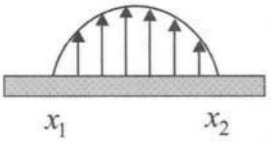
Load_Spec = $[2 \quad x_1 \quad x_2 \quad q_1 \quad q_2]$

(D3) Quadratic

$f(x) = c_0 + c_1 x + c_2 x^2 \quad \text{for } x_1 \le x \le x_2$

Load_Spec = $[3 \quad x_1 \quad x_2 \quad c_0 \quad c_1 \quad c_2]$

(D4) Sinusoidal



$f(x) = A\sin(\omega x + \phi) \quad \text{for } x_1 \le x \le x_2$

Load_Spec = $[4 \quad x_1 \quad x_2 \quad A \quad \omega \quad \phi]$, ϕ in radians

2.6 References

1. Bedford A, Liechti K M 2000 *Mechanics of Materials*, Prentice-Hall: Upper Saddle River, New Jersey.
2. Gere J M, Timoshenko S P 1997 *Mechanics of Materials*, 4th ed., PWS Publishing Co.: Boston.
3. Popov E P 1998 *Engineering Mechanics of Solids*, 2nd ed., Prentice-Hall: Upper Saddle River, New Jersey.
4. Riley W F, Sturges L D, Morris D H 1999 *Mechanics of Materials*, 5th ed., John Wiley & Sons, Inc.: New York.
5. Shames I H, Pitarresi J M 2000 *Introduction to Solid Mechanics*, 3rd ed., Prentice-Hall: Upper Saddle River, New Jersey.

This Page Intentionally Left Blank

3 Static Analysis of Bars, Shafts, and Strings

Inside
- Getting Started
- System Description
- Static Analysis by the Toolbox
- Stepped Bars and Shafts
- The Distributed Transfer Function Method
- Quick Solution Guide
- References

3.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for static analysis of longitudinal bars, torsional shafts, and taut strings.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB, and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **`TBdemo`** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

Quick Tutorial

To show how to use the Toolbox, consider a circular torsional shaft of torsional rigidity GJ and length L in Fig. 3.1.1, which is connected to a torsional spring of coefficient k at the left end, and fixed at the right end. The shaft is subject to a uniformly distributed torque $\tau(x) = \tau_0$ in region $a \leq x \leq b$. Let the values of the system and load parameters be

$$GJ = 245 \text{ N} \cdot \text{m}^2, \quad L = 5 \text{ m}$$
$$k = 210 \text{ N} \cdot \text{m/rad}$$
$$\tau_0 = 25 \text{ N}, \quad a = 1.5 \text{ m}, \quad b = 3.5 \text{ m}$$

To obtain the rotation $\theta(x)$ and internal torque $T(x)$ of the shaft, in the MATLAB command window, type:

```
>> GJ=245; L=5; k=210;
>> setshaft(GJ, L, [3 k, 1])
>> tau0=25; a=1.5; b=3.5;
>> force2os([1 a b tau0 0]);
```

where >> is the prompt in the MATLAB command window. This yields the spatial distribution of the rotation and internal torque plotted in Fig. 3.1.2.

In the above static analysis, two functions from the Toolbox have been called:

(a) Function **setbar** that specifies the shaft parameters and boundary conditions.
(b) Function **force2os** that computes the response of the shaft under the torque.

These functions, along with others, will be introduced in the subsequent sections.

For a quick solution, the reader may refer directly to the Quick Solution Guide (Section 3.6), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To better understand how the toolbox works, the reader is encouraged to go through the entire chapter. For further reading on the subject refer to Section 3.7.

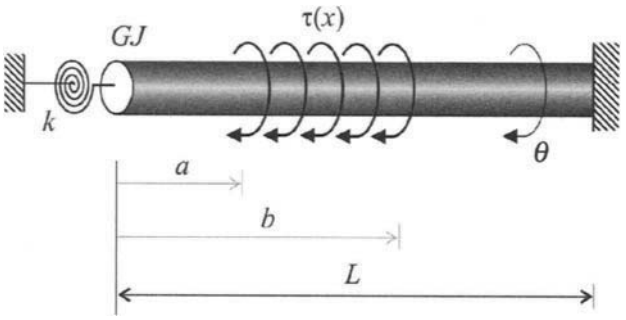


FIGURE 3.1.1 A circular torsional shaft under a uniformly distributed torque.

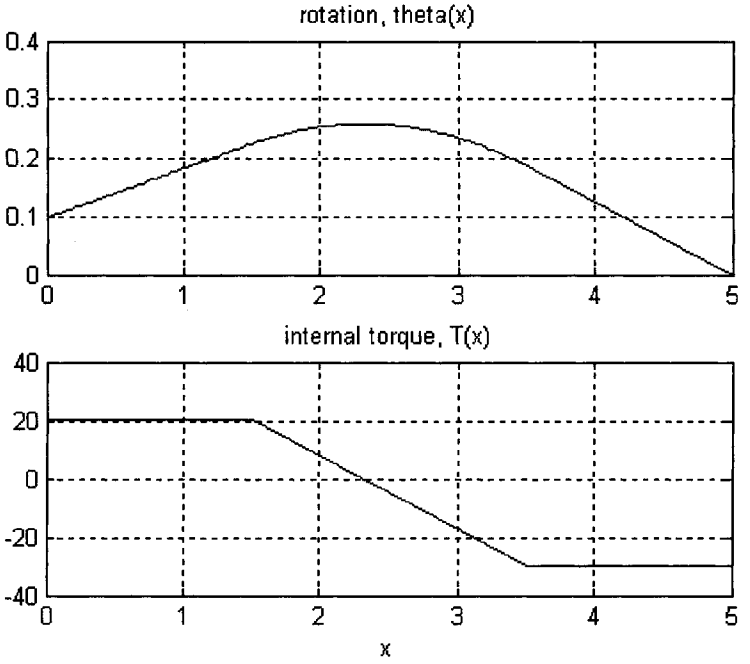


FIGURE 3.1.2 Shaft response distribution.

3.2 System Description

This chapter is concerned with the displacement and internal force of the following three types of uniformly distributed systems:

- Elastic bars in longitudinal deformation;
- Circular shafts in torsional deformation; and
- Taut strings with lateral deflection.

The description of these systems is given in Tables 3.2.1 to 3.2.3.

Note. The torsional rigidity of a circular torsional shaft is a product of the shear modulus G and the polar moment of area J of the cross section of the shaft. The polar moment of an area A is defined by

$$J = \int_A r^2 dA.$$

For a solid circular cross section of diameter D,

$$J = \frac{1}{32}\pi D^4.$$

In this book, bars, shafts, and strings shall be called second-order distributed systems as their displacements are governed by second-order differential equations. The governing equations and boundary conditions of these three types of systems can be cast into the

TABLE 3.2.1 Longitudinal Bars

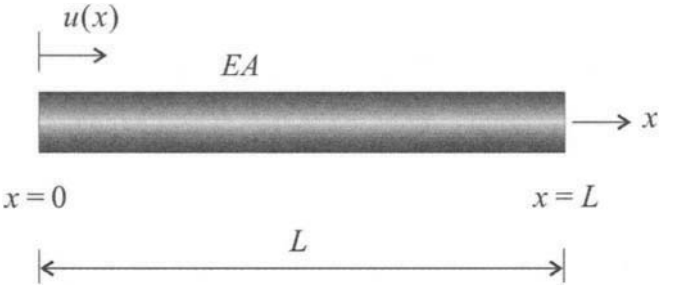
Governing equation

$$-EA\frac{d^2}{dx^2}u(x) = q(x)$$

$u(x)$: longitudinal displacement

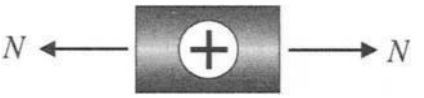
$q(x)$: external load

EA: longitudinal rigidity



Internal force (axial force)

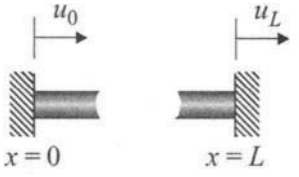
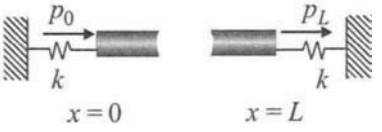
$$N(x) = EA\frac{d}{dx}u(x)$$



Sign convention

Boundary conditions and boundary disturbances

(u_0, u_L - boundary displacements; p_0, p_L - external loads; k - spring coefficient)

(B1) Fixed end	(B2) Free end	(B3) End spring
		
Left: $u(0) = u_0$	Left: $-EA\frac{d}{dx}u(0) = p_0$	Left: $ku(0) - EA\frac{d}{dx}u(0) = p_0$
Right: $u(L) = u_L$	Right: $EA\frac{d}{dx}u(L) = p_L$	Right: $ku(L) + EA\frac{d}{dx}u(L) = p_L$

following unified form:

Governing equation:

$$-\alpha_{ST}\frac{d^2}{dx^2}w(x) = f(x), \quad 0 \le x \le L \tag{3.1}$$

where $w(x)$ is a general displacement, which is $u(x)$ for a bar, $\theta(x)$ for a shaft, and $y(x)$ for a string; α_{ST} is a stiffness parameter, which is EA for a bar, GJ for a shaft, and T for a string.

Internal force:

$$\sigma(x) = \alpha_{ST}\frac{d}{dx}w(x) \tag{3.2}$$

Here $\sigma(x)$ is N for a bar, T for a shaft, and T_y for a string.

TABLE 3.2.2 Circular Torsional Shafts

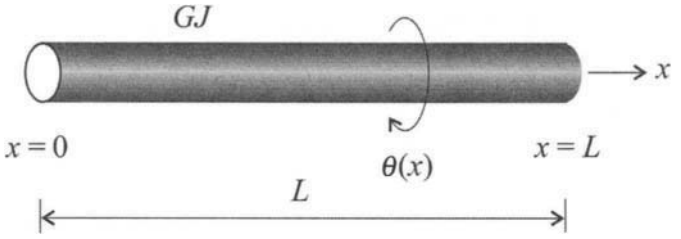
Governing equation

$$-GJ\frac{d^2}{dx^2}\theta(x) = \tau(x)$$

$\theta(x)$: rotation (twist)

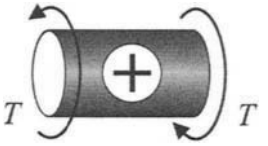
$\tau(x)$: external torque

GJ: torsional rigidity



Internal force (torque)

$$T(x) = GJ\frac{d}{dx}\theta(x)$$

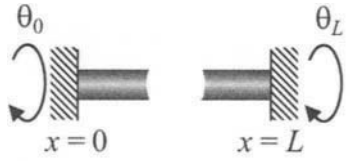


Sign convention

Boundary conditions and boundary disturbances

(θ_0, θ_L - boundary rotations; T_0, T_L - external torques; k - spring coefficient)

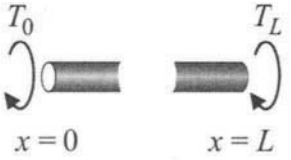
(B1) Fixed end



Left: $\theta(0) = \theta_0$

Right: $\theta(L) = \theta_L$

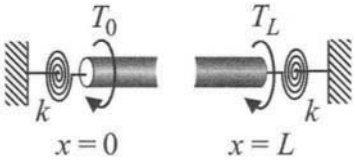
(B2) Free end



Left: $-GJ\frac{d}{dx}\theta(0) = T_0$

Right: $GJ\frac{d}{dx}\theta(L) = T_L$

(B3) End spring



Left: $k\theta(0) - GJ\frac{d}{dx}\theta(0) = T_0$

Right: $k\theta(L) + GJ\frac{d}{dx}\theta(L) = T_L$

Boundary conditions:
(B1) Fixed end:

$$\begin{aligned} &\text{Left end } (x = 0), \quad w(0) = w_0 \\ &\text{Right end } (x = L), \quad w(L) = w_L \end{aligned} \tag{3.3a}$$

(B2) Free end:

$$\begin{aligned} &\text{Left end } (x = 0), \quad -\alpha_{ST}\tfrac{d}{dx}w(0) = p_0 \\ &\text{Right end } (x = L), \quad \alpha_{ST}\tfrac{d}{dx}w(L) = p_L \end{aligned} \tag{3.3b}$$

TABLE 3.2.3 Taut Strings

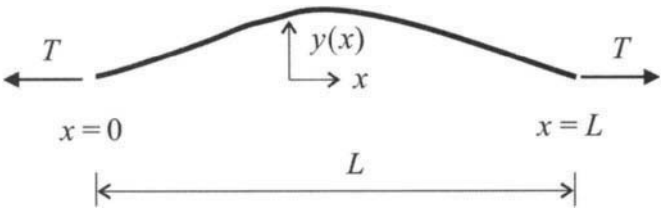
Governing equation

$$-T\frac{d^2}{dx^2}y(x) = p(x)$$

$y(x)$: lateral deflection

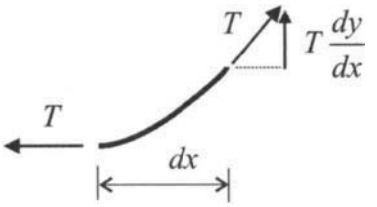
$p(x)$: lateral external load

T: string tension



Internal force (lateral projection of tension)

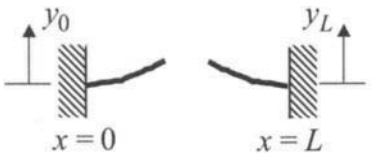
$$T_y = T\frac{d}{dx}y(x)$$



Boundary conditions and boundary disturbances

(y_0, y_L - boundary displacements; p_0, p_L - external loads; k - spring coefficient)

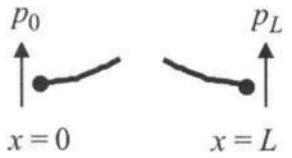
(B1) Fixed end



Left: $y(0) = y_0$

Right: $y(L) = y_L$

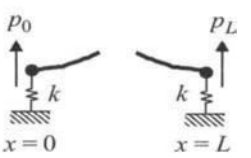
(B2) Free end



Left: $-T\frac{d}{dx}y(0) = p_0$

Right: $T\frac{d}{dx}y(L) = p_L$

(B3) End spring



Left: $ky(0) - T\frac{d}{dx}y(0) = p_0$

Right: $ky(L) + T\frac{d}{dx}y(L) = p_L$

(B3) End spring:

$$\begin{aligned} &\text{Left end } (x = 0), \quad kw(0) - \alpha_{ST}\tfrac{d}{dx}w(0) = p_0 \\ &\text{Right end } (x = L), \quad w(L) + \alpha_{ST}\tfrac{d}{dx}w(L) = p_L \end{aligned} \tag{3.3c}$$

In the above equations, w_0 and w_L are boundary displacements; p_0 and p_L are external loads; and k is a spring coefficient.

A fundamental problem of second-order distributed systems is to solve Equations (3.1) and (3.3) for displacement $w(x)$ and internal force $\sigma(x)$. Refer to Section 3.5 for an analytical solution method for this static problem.

3.3 Static Analysis by the Toolbox

The Toolbox of this chapter provides several MATLAB functions for determining exact static response of second-order distributed systems, subject to external loads, and boundary disturbances. The solution procedure takes the following three steps:

Step 1. Set up system parameters and boundary conditions by one of the functions **setbar**, **setshaft**, and **setstring**.
Step 2. Determine system response by functions **force2os** and **bd2os**.
Step 3. Plot computed response by function **plot2os**.

These functions shall be presented as follows.

3.3.1 System Setup

In static analysis of a second-order distributed system, one of the functions `setbar`, `setshaft`, and `setstring` will be called; see Window 3.1. These functions have to be executed before functions `force2os`, `bd2os` or `plot2os` can be used.

Window 3.1. Functions `setbar`, `setshaft`, `setstring`

MATLAB Functions: **`setbar`, `setshaft`, `setstring`**

Purpose:
To set up parameters and boundary conditions for a second-order distributed system.

Synopsis:

```
setbar(EA, L, BC_Spec, n)
setshaft(GJ, L, BC_Spec, n)
setstring(T, L, BC_Spec, n)
```

Description:
`setbar(EA, L, BC_Spec, n)` undertakes the following tasks:

(a) It inputs stiffness `EA` and length `L` for a bar.
(b) It specifies the boundary conditions of the bar by a vector `BC_Spec`, whose specification is given in Table 3.3.1.
(c) It sets up the number `n` of points along the bar length for simulation. By default, `n` = 201.
(d) It displays the information on the above system setup.

Functions `setshaft` and `setstring` undertake similar tasks, for a shaft and a string, respectively.

As can be seen from Table 3.3.1, the boundary conditions of a second-order system are assigned with a row vector

$$\texttt{BC\_Spec} = [\texttt{BC\_Left} \quad \texttt{BC\_Right}] \tag{3.4}$$

TABLE 3.3.1 Boundary Conditions and Boundary Disturbances

Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`
Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

Boundary Conditions	*Left Boundary (x = 0)*		*Right Boundary (x = L)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B1) Fixed end at $x = 0$, $w(0) = w_0$ at $x = L$, $w(L) = w_L$	[1]	$[w_0]$	[1]	$[w_L]$
(B2) Free end at $x = 0$, $-\alpha_{ST}\frac{d}{dx}w(0) = p_0$ at $x = L$, $\alpha_{ST}\frac{d}{dx}w(L) = p_L$	[2]	$[p_0]$	[2]	$[p_L]$
(B3) End spring at $x = 0$, $kw(0) - \alpha_{ST}\frac{d}{dx}w(0) = p_0$ at $x = L$, $w(L) + \alpha_{ST}\frac{d}{dx}w(L) = p_L$	[3 k]	$[p_0]$	[3 k]	$[p_L]$

which has two subvectors `BC_Left` and `BC_Right` specifying the boundary conditions at the left end ($x = 0$) and the right end ($x = L$), respectively. As an example, vector `BC_Spec` is assigned in the following three cases of boundary conditions:

Case (a) Fixed-fixed (B1-B1): `BC_Spec = [1 1]`
Case (b) Fixed-free (B1-B2): `BC_Spec = [1 2]`
Case (c) Spring end-fixed (B3-B1): `BC_Spec = [3 k 2]`

Once function `setbar`, or `setshaft`, or `setstring` is called, the information of the specified can be accessed at any time by typing the command

```
>> systinfo
```

3.3.2 Response to External Forces

The Toolbox of this chapter provides a function `force2os` for computing the static response of a second-order distributed system under external forces; see Windows 3.2.

Window 3.2. Function `force2os`

MATLAB Function: **`force2os`**

Purpose:
To plot the static response of a second-order system under external forces.

Window 3.2. Function force2os (*Continued*)

Synopsis:

```
force2os(Load_Spec)
[y, x] = force2os(Load_Spec)
```

Description:

`force2os(Load_Spec)` plots the response of the system subject to `Nf` external loads that are specified by an `Nf`×5matrix

$$\texttt{Load\_Spec} = \begin{bmatrix} \texttt{Load\_1} \\ \texttt{Load\_2} \\ \vdots \\ \texttt{Load\_Nf} \end{bmatrix}$$

where each row of the matrix specifies an external load. A typical row is of the form

```
Load_j = [load_type p1 p2 p3 p4]
```

where `load_type` is an integer from 0 to 2 identifying one of the three types of external loads given in Table 3.3.2, and `p1` to `p4` are parameters describing the location and pattern of the load.

`[y, x] = force2os(Load_Spec)` returns the computed response in a two-row matrix `y`, and spatial points of simulation in a vector `x`. With `y` and `x`, the system response can be plotted by the MATLAB function `plot` as follows:

```
plot(x, y(1,:))   plot the spatial distribution of displacement
plot(x, y(2,:))   plot the spatial distribution of internal force
```

Also, the system response can be plotted by function `plot2os(y)`; see Window 3.4.

Table 3.3.2 lists three types of external loads considered in the Toolbox. The first element of the vector `Load_j` is assigned as follows:

```
load_type = 0    for a pointwise force (L1);
load_type = 1    for a uniform load (D1);
load_type = 2    for linearly distributed load (D2).
```

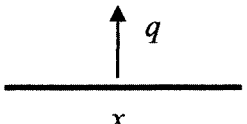
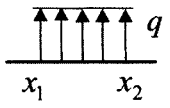
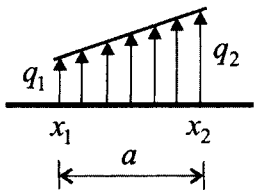
The x_q, x_τ, x_1, and x_2 are the coordinate parameters measured from the left end of the system. These parameters describe the location or region of action of a load. In addition, $\delta(\cdot)$ in L1 is the Dirac delta function.

EXAMPLE 3.1

Consider a fixed-fixed string of tension $T = 1{,}500\ N$ and length $L = 2\ m$, see Fig. 3.3.1. The string carries three lumped masses:

$$m_1 = 10\ kg \quad \text{at} \quad x_1 = 0.45\ m$$
$$m_2 = 15\ kg \quad \text{at} \quad x_2 = 0.92\ m$$
$$m_3 = 12\ kg \quad \text{at} \quad x_3 = 1.36\ m$$

TABLE 3.3.2 Specification of External Loads

Load	*Load Specification Vector*
(L1) Pointwise force  $f(x) = q\delta(x - x_q)$	`Load_j = [0 xq q 0 0]`
(D1) Uniform distribution  $f(x) = q$ for $x_1 \le x \le x_2$	`Load_j = [1 x1 x2 q 0 0]`
(D2) Linear distribution  $f(x) = q_1 + \frac{q_2 - q_1}{a}(x - x_1)$ for $x_1 \le x \le x_2$	`Load_j = [2 x1 x2 q1 q2]`

Our task is to determine the profile of string lateral deflection under the weights $W_i = -m_i g$ (pointwise forces), where $g = 9.81$ m/s^2. The following MATLAB commands

```
>> T=1500; L=2; BC_Spec=[1 1];
>> setstring(T, L, BC_Spec)
>> g=9.81;
>> Load_Spec=[0 0.45 -10*g 0 0; 0 0.92 -15*g 0 0; 0 1.36 -12*g 0 0];
>> force2os(Load_Spec)
```

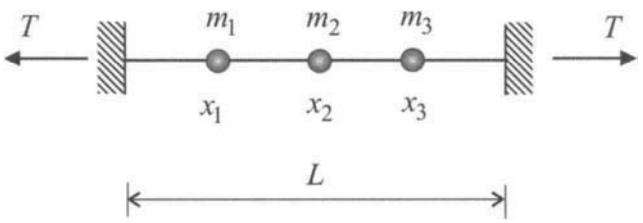


FIGURE 3.3.1

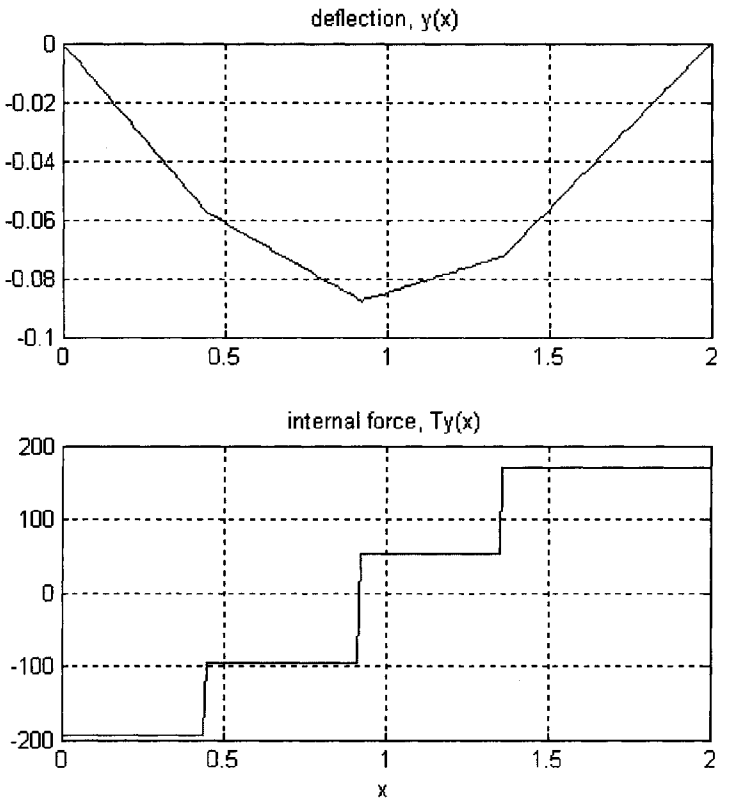


FIGURE 3.3.2

yield the spatial distribution of the deflection y and projected tension T_y shown in Fig. 3.3.2.

3.3.3 Response to Boundary Disturbances

To compute system response subject to prescribed boundary disturbances (displacements and forces), function `bd2os` from the Toolbox can be used; see Window 3.3.

Window 3.3. Function bd2os

MATLAB Function: **bd2os**

Purpose:
To plot static response of a second-order system under boundary disturbances.

Synopsis:

```
bd2os(BD_Spec)
[y, x] = bd2os(BD_Spec)
```

Window 3.3. Function bd2os (*Continued*)

Description:
`bd2os(BD_Spec)` plots the response of the system subject to boundary disturbances that are specified by a row vector

```
BD_Spec = [BD_Left BD_Right]
```

where the subvectors `BD_Left` and `BD_Right` describe the disturbances at the left end ($x = 0$) and right end ($x = L$) of the system, respectively; see Table 3.3.1.

`[y, x] = bd2os(BD_Spec)` returns the computed response in a two-row matrix `y`, and spatial points of simulation in a vector `x`. With `y` and `x`, the system response can be plotted by the MATLAB function `plot` as follows:

`plot(x, y(1,:))`	plot the spatial distribution of displacement
`plot(x, y(2,:))`	plot the spatial distribution of internal force

Also, the system response can be plotted by function `plot2os(y)`; see Window 3.4.

EXAMPLE 3.2

The circular shaft in Fig. 3.3.3 has torsional rigidity $GJ = 200$ and length $L = 10$, and is connected to a torsional spring of $k = 30$ at its left end and fixed at its right end. An end torque $T_0 = 25$ is applied to the left end. The boundary disturbance vector by Table 3.3.1 is

```
BD_Spec = [25 0]
```

By the commands

```
>> setshaft(200, 10, [3 30 1])
>> BD_Spec = [25 0];
>> bd2os(BD_Spec)
```

the static rotation (twist) and internal torque of the shaft is plotted in Fig. 3.3.4.

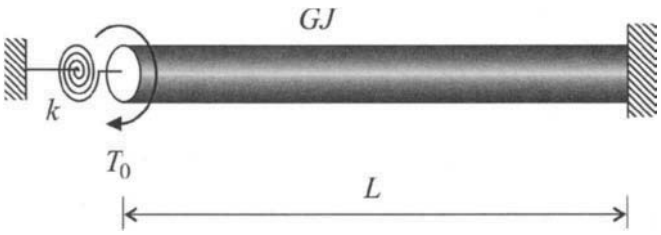


FIGURE 3.3.3

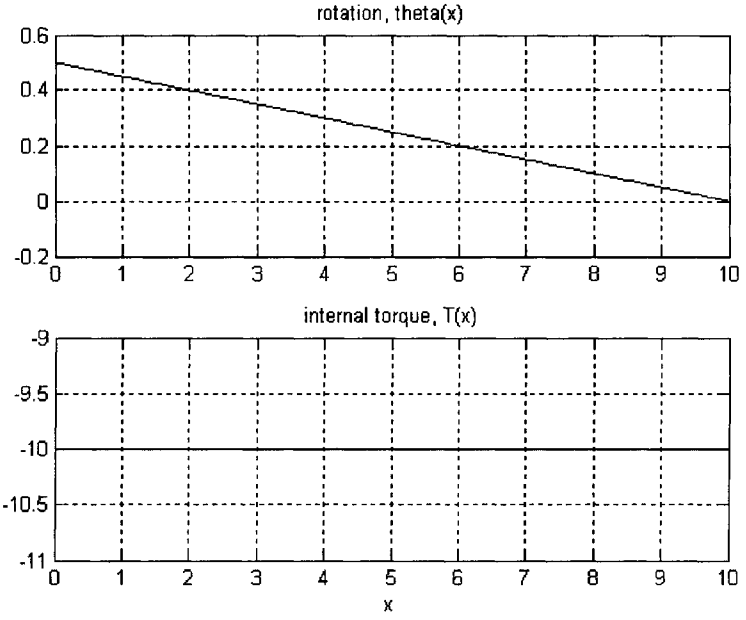


FIGURE 3.3.4

3.3.4 Total Response

When a second-order system is subject to both external loads and boundary disturbances, the total static response of the system can be obtained by the principle of superposition:

```
y = force2os(Load_Spec) + bd2os(BD_Spec);
plot2os(y)
```

where functions `force2os` and `bd2os` have been introduced in the previous sections and `plot2os` is a plotting function given in Window 3.4.

Window 3.4. Function `plot2os`

MATLAB Function: **`plot2os`**

Purpose:
To plot static response of a second-order system.

Synopsis:

```
plot2os(y)
plot2os(y, opt)
```

Description:
`plot2os(y)` plots spatial distributions of the static displacement and internal force of a second-order system. Here `y` is a matrix of system response data that is obtained by function `force2os` or `bd2os`.

Window 3.4. Function `plot2os` (*Continued*)

`plot2os(y, opt)` plots the response with the following options:

`opt = 1`	plot spatial distribution of displacement only
`opt = 2`	plot spatial distribution of internal force only

EXAMPLE 3.3

In Fig. 3.3.5, a longitudinal bar is fixed at its left end and connected at its right end to a spring of $k = 65$. The bar parameters are $EA = 250$ and $L = 1$. There are two external loads: a pointwise force with $f_0 = 2.7$ applied at $x_f = 0.4$ and a linearly distributed load (Type D2 in Table 3.3.2) with $q = 4$. The bar is also subject to a boundary displacement at its left end, $u_0 = 0.02$. The static response of the bar is plotted in Fig. 3.3.6 by

```
>> setbar(250, 1, [1 3 65])
>> loads = [0 0.4 -2.7 0 0; 2 0 1 4 0];
>> bd = [0.02 0];
>> y = force2os(loads) + bd2os(bd);
>> plot2os(y)
```

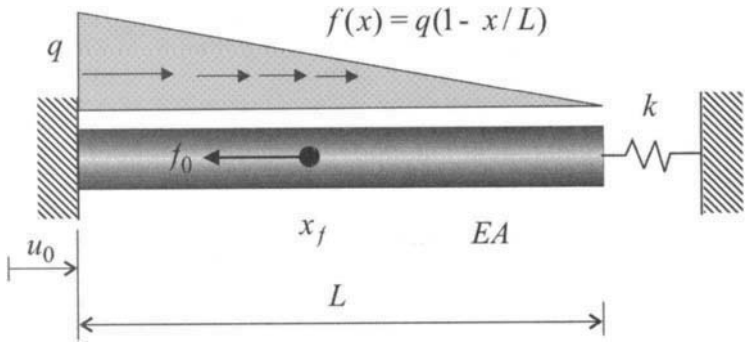


FIGURE 3.3.5

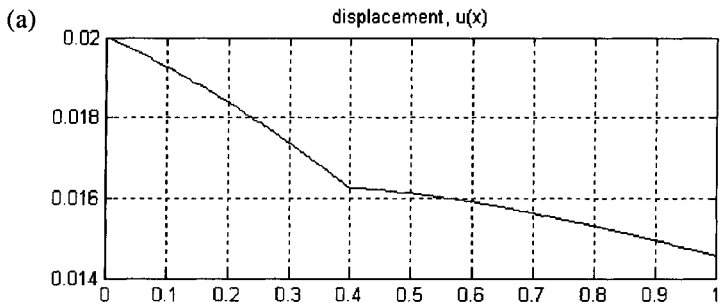


FIGURE 3.3.6

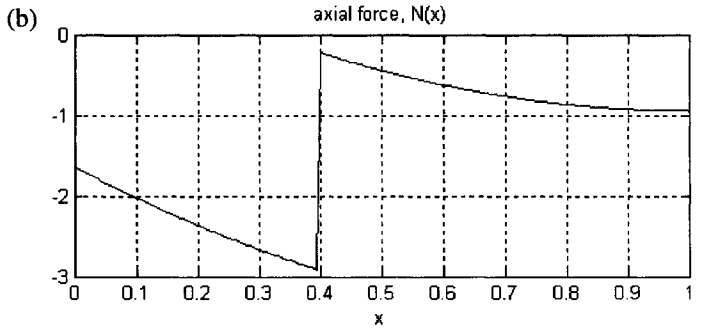


FIGURE 3.3.6 *Continued.*

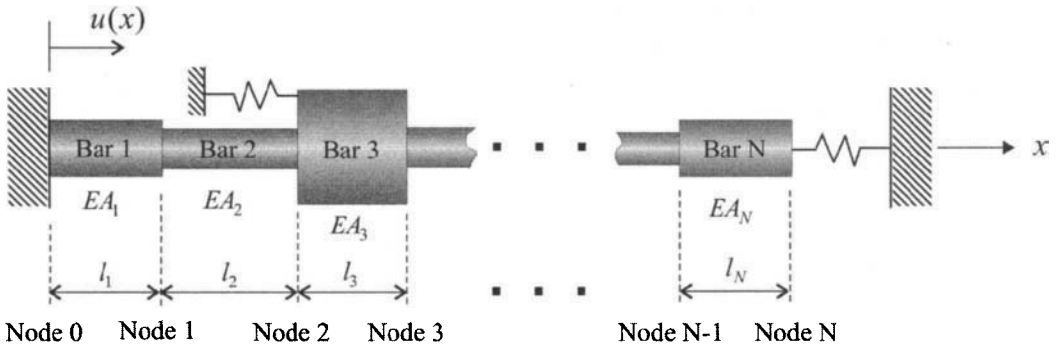


FIGURE 3.3.7 Schematic of a stepped bar.

3.4 Stepped Bars and Shafts

This section is concerned with static analysis of stepped bars and shafts. Description of such distributed systems is given in Section 3.4.1, and MATLAB functions for static analysis are presented in Section 3.4.2.

3.4.1 System Description

Stepped Bars Figure 3.4.1 shows the schematic of a stepped bar, where the structure is an assembly of N uniform bar elements that are serially connected at N-1 points called *nodes*. At those nodes, there may be spring constrains, as the one at node 2. The left and right ends of the structure are labeled as Nodes 0 and N.

The longitudinal displacement $u_i(x)$ of the ith element is described by

$$-EA_i \frac{d^2}{dx^2} u_i(x) = q(x), \quad 0 \le x \le l_i \tag{3.5}$$

where EA_i and l_i are the longitudinal rigidity and length of the element, respectively, and x is a local coordinate. The boundary conditions at Node 0 (the left end of the first element) and Node N (the right end of the Nth element) are specified according to Table 3.3.1. The connection of the ith and $(i + 1)$th bar elements at node i satisfies the following compatibility

conditions

$$u_i(l_i) = u_{i+1}(0); \quad EA_i \frac{d}{dx} u_i(l_i) + k_c u_i(l_i) = EA_{i+1} \frac{d}{dx} u_{i+1}(0) \tag{3.6}$$

where k_c is the coefficient of a spring that is imposed to the node as a constraint.

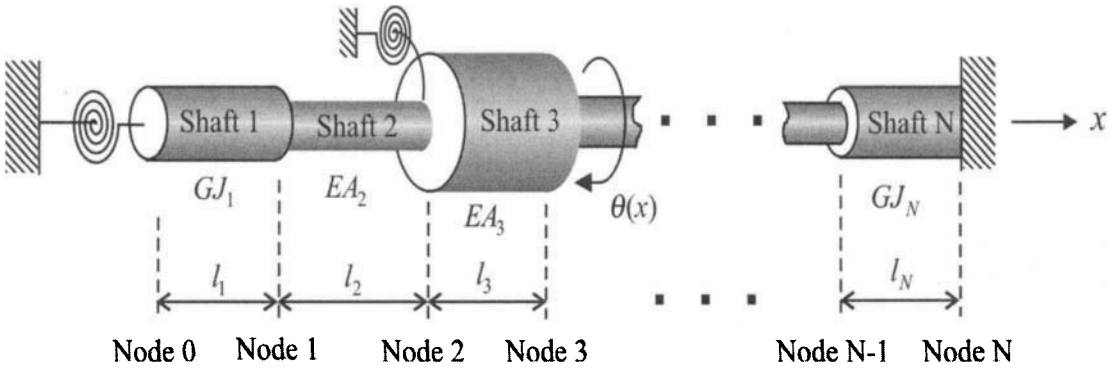


FIGURE 3.4.1 Schematic of a stepped shaft.

Stepped Shafts A stepped torsional shaft is an assembly of N uniform circular shaft elements connected at N-1 nodes, as shown in Fig 3.4.2. The rotational displacement $\theta_i(x)$ of the ith element is governed by

$$-GJ_i \frac{d^2}{dx^2} \theta_i(x) = \tau(x), \quad 0 \le x \le l_i \tag{3.7}$$

where GJ_i and l_i are the torsional rigidity and length of the element, respectively, and x is a local coordinate. The boundary conditions at Node 0 (the left end) and Node N (the right end) of the structure are assigned according to Table 3.3.1. The connection of the ith and $(i + 1)$th shaft elements at node i satisfies the following compatibility conditions

$$\theta_i(l_i) = \theta_{i+1}(0); \quad GJ_i \frac{d}{dx} \theta_i(l_i) + k_c \theta_i(l_i) = GJ_{i+1} \frac{d}{dx} \theta_{i+1}(0) \tag{3.8}$$

where k_c is the coefficient of a spring that is imposed to the node as a constraint; see Node 2 of Fig. 3.4.2 for instance.

As can be seen, stepped bars and shafts have the same type of governing equations.

Static Problem A basic problem is to determine the static response (displacement and internal force) by solving Eqs. (3.5) and (3.6) for a stepped bar, or solving Eqs. (3.7) and (3.8) for a stepped shaft.

3.4.2 Solution by the Toolbox

System Setup As in analysis of single-body bars and shafts, one needs to set up a stepped structure first before static response can be computed. This is done through use of functions `setbar2` and `setshaft2`, as given in Window 4.1.

Window 4.1. Functions setbar2, setshaft2

MATLAB Functions: **setbar2, setshaft2**

Purpose:
To assign parameters, boundary conditions, and constraints for a stepped bar or shaft.

Synopsis:

```
setbar2(Element_Spec, BC_Spec, Constr_Spec, n)
setshaft2(Element_Spec, BC_Spec, Constr_Spec, n)
```

Description:
`setbar2(Element_Spec, BC_Spec, Constr_Spec, n)` specifies a stepped bar by undertaking the following tasks:

(a) It inputs the stiffness and length of the bar elements by a matrix `Element_Spec`;
(b) It specifies the boundary conditions of the structure by a vector `BC_Spec`;
(c) It imposes spring constraints onto the structure by a matrix `Constr_Spec`; and
(d) It sets up `n` equally-spaced points along each element for computation. By default, `n` = 101.

If the structure has no spring constraints, simply use

```
setbar2(Element_Spec, BC_Spec)
```

or

```
setshaft2(Element_Spec, BC_Spec, 0, n)
```

`setshaft2(Element_Spec, BC_Spec, Constr_Spec, n)` undertakes similar tasks as function `setbar2`, except for a stepped shaft.

Note. In Window 4.1, function `setbar2` or `setshaft2` assigns the parameters, boundary conditions, and constraints for an N-element stepped structure by the following input variables:

(i) The parameters of the N elements are assigned by an $N \times 2$ matrix, which for a stepped bar is

$$\texttt{Element\_Spec} = \begin{bmatrix} EA_1 & l_1 \\ EA_2 & l_2 \\ \vdots & \vdots \\ EA_N & l_N \end{bmatrix} \tag{3.9a}$$

and which for a stepped shaft is

$$\texttt{Element\_Spec} = \begin{bmatrix} GJ_1 & l_1 \\ GJ_2 & l_2 \\ \vdots & \vdots \\ GJ_N & l_N \end{bmatrix} \tag{3.9b}$$

Here, the ith row of the matrix contains the stiffness and length of the ith element.

(ii) The boundary conditions of the structure (at Nodes 0 and N) are assigned with a row vector

$$\texttt{BC\_Spec} = [\texttt{BC\_Left} \quad \texttt{BC\_Right}] \tag{3.10}$$

where `BC_Left` and `BC_Right` are two row subvectors describing the boundary conditions at the left end (Node 0) and the right end (Node N) of the structure, respectively. The formation of `BC_Left` and `BC_Right` follows Table 3.3.1, for three basic types of boundary conditions.

(iii) The spring constraints at any of the interior nodes (1,2,...,N-1) are specified by an $N_c \times 2$ matrix

$$\texttt{Constr\_Spec} = \begin{bmatrix} \texttt{Constr\_1} \\ \texttt{Constr\_2} \\ \vdots \\ \texttt{Constr\_Nc} \end{bmatrix} \tag{3.11}$$

where `Nc` is the total number of constraints. The jth row of the matrix specifies the jth spring constraint, and is of the form

```
Constr_j = [node_no kc]
```

where `node_no` is the node number, and `kc` is the spring coefficient. For a stepped structure with no constraints, assign `Constr_Spec = 0` or `Constr_Spec = [ ]`.

Static Response This Toolbox provides a function `step2osfb` for computing the static response (displacement and internal force) of a stepped bar or shaft subject to external forces and boundary disturbances; see Windows 4.2.

Window 4.2. Function `step2osfb`

MATLAB Function: **`step2osfb`**

Purpose:
To compute and display the static response of a stepped second-order system (bar or shaft) under external forces and/or boundary disturbances.

Synopsis:

```
step2osfb(Load_Spec, BD_Spec)
[Resp, x] = step2osfb(Load_Spec, BD_Spec)
```

Description:
`step2osfb(Load_Spec, BD_Spec)` plots the displacement and internal force of the stepped structure subject to external loads specified by a matrix `Load_Spec` and boundary disturbances specified by a vector `BD_Spec`. The function also displays the static response at all the nodes of the structure. The formation of `Load_Spec` and `BD_Spec` is given after this window.

If the stepped structure is only subject to external loads, simply use

```
step2osfb(Load_Spec)
```

Window 4.2. Function step2osfb (*Continued*)

On the other hand, if the structure is only subject to boundary disturbances, use

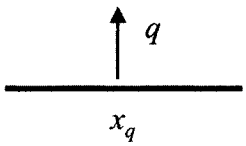
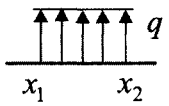
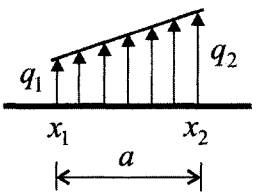
```
step2osfb(0, BD_Spec)
```

`[Resp, x] = step2osfb(Load_Spec, BD_Spec)` returns the computed response in a two-row matrix `Resp` and the spatial points of simulation in a vector `x`. With `Resp` and `x`, the spatial distributions of the static response can be plotted by the MATLAB function `plot` as follows:

```
plot(x, Resp(1,:))    for displacement
plot(x, Resp(1,:))    for internal force
```

Also, function `plot2os2` from this toolbox can be used to plot spatial distributions of the static response; see Window 4.3.

TABLE 3.4.1 Specification of External Loads

Load	*Load Specification Vector*
(L1) Pointwise force  $f(x) = q\delta(x - x_q)$	`Load_j = [element_no 0` x_q `q 0 0]`
(D1) Uniform distribution  $f(x) = q$ for $x_1 \le x \le x_2$	`Load_j = [element_no 1` x_1 x_2 `q 0 0]`
(D2) Linear distribution  $f(x) = q_1 + \frac{q_2 - q_1}{a}(x - x_1)$ for $x_1 \le x \le x_2$	`Load_j = [element_no 2` x_1 x_2 q_1 q_2`]`

The Load_Spec and BD_Spec in Table 3.4.1 are specified as follows. The formation of vector BD_Spec follows Table 3.3.1. Matrix Load_Spec is of the form

$$\texttt{Load\_Spec} = \begin{bmatrix} \texttt{Load\_1} \\ \texttt{Load\_2} \\ \vdots \\ \texttt{Load\_Nf} \end{bmatrix}$$

Each row of the matrix specifies an external load by

```
Load_j = [element_no load_type p1 p2 p3 p4]
```

where `element_no` is an element number indicating which element the load is applied to, `load_type` is an integer from 0 to 2 identifying one of the three types of external loads given in Table 3.4.1, and `p1` to `p4` are parameters describing the location and form of the load.

EXAMPLE 4.1

In Fig. 3.4.3, a two-step torsional shaft is fixed at the left end and is constrained by two springs of coefficients k_L and k_c, respectively. The bar is subject to a pointwise torque on the first shaft element at location x_τ and a uniform torque τ_0 on part of the second element, and experiences a rotation θ_0 at the left end. Let the system parameters be

$$GJ_1 = 250, \quad l_1 = 0.4$$
$$GJ_2 = 500, \quad l_2 = 0.6$$
$$k_c = 320, \quad k_L = 650$$

Assume that the load parameters are

$$\tau_0 = 25, \quad x_\tau = 0.2; \quad q = 8.5, \quad a = 0.3$$

By

```
>> Ele_Spec = [250 0.4; 500 0.6];
>> setshaft2(Ele_Spec, [1 3 650], [1 320]);
>> load =[1 0 0.2 25 0 0; 2 1 0.3 0.6 8.5 0];
>> step2osfb(load, [0.01 0])
```

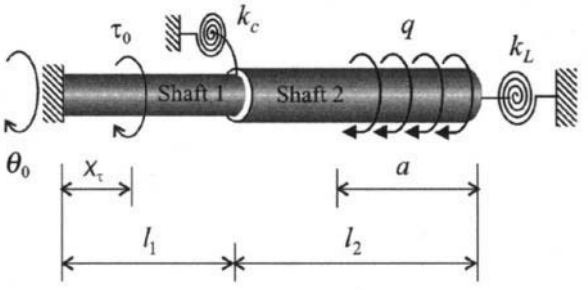


FIGURE 3.4.2

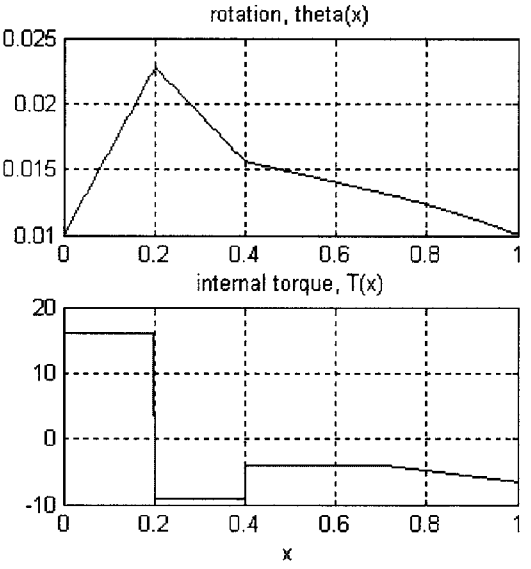


FIGURE 3.4.3

the static response of the stepped shaft is plotted in Fig. 3.4.4. Also, function `step2osfb` yields the following information about the response at the nodes:

```
Response at nodes:
 at node 0 (left end):  theta = 0.01, T = 16.0111
 at node 1:             theta = 0.015618, T(-0) = -8.9889, T(+0) = -3.9912
 at node 2 (right end): theta = 0.010063, T = -6.5412
```

Some Utility Functions Besides `setbar2`, `setshaft2`, and `step2osfb`, the Toolbox also has the following utility functions:

(i) Function **`systinfo2`.** The command

```
>> systinfo2
```

displays the information of a stepped bar or shaft that is set up by `setbar2` or `setshaft2`.

(ii) Function **`plot2os2`** for plotting the spatial distribution of the static response of a stepped bar or shaft, as presented in Window 4.3.

Window 4.3. Function `plot2os2`

MATLAB Function: **`plot2os2`**

Purpose:
To plot static response of a stepped bar or shaft.

Synopsis:

```
plot2os2(y)
plot2os2(y, opt)
```

Window 4.3. Function `plot2os2` (*Continued*)

Description:
`plot2os2(y)` plots spatial distributions of the static displacement and internal force of a stepped bar or shaft. Here `y` is a matrix of system response data that is obtained by function `step2osfb`.

`plot2os2(y, opt)` plots the static response with the following options:

`opt` = 1 plot spatial distribution of displacement only
`opt` = 2 plot spatial distribution of internal force only

EXAMPLE 4.2

Consider the same shaft with the same loads and boundary displacement as given in Example 4.1. The spatial distribution of the internal force of the shaft is plotted in Fig. 3.4.5 by

```
>> setshaft2([250 0.4; 500 0.6], [1 3 650], [1 320])
>> load =[1 0 0.2 25 0 0; 2 1 0.3 0.6 8.5 0];
>> y = step2osfb(load, [0.01 0]);
>> plot2os2(y, 2)
```

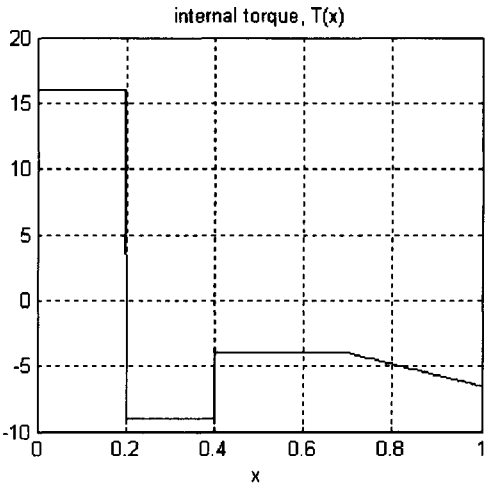


FIGURE 3.4.4

3.5 The Distributed Transfer Function Method

The Toolbox of this chapter is developed based on the Distributed Transfer Function Method (DTFM), which is an exact analytical modeling and solution method for many distributed systems including bars, shafts, strings, and beams. This section presents the application of

the DTFM in second-order distributed systems. The reader who is interested in knowing more about the method may refer to Appendix C.

State-Space Formulation

The governing equation (3.1) is cast into an equivalent state-space form

$$\frac{d}{dx}\{\eta(x)\} = [F]\,\eta(x) - \frac{1}{\alpha_{ST}} f(x)\cdot\{e_2\}, \quad 0 \le x \le L \tag{3.12}$$

where

$$\{\eta(x)\} = \begin{pmatrix} w(x) \\ \dfrac{d}{dx}w(x) \end{pmatrix}, \quad [F] = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \{e_2\} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{3.13}$$

The boundary conditions can also be written in terms of the state-space vector $\{\eta\}$:

$$[M_b]\,\{\eta(0)\} + [N_b]\,\{\eta(L)\} = \{\gamma_b\} \tag{3.14}$$

where $[M_b]$ and $[N_b]$ are two-by-two boundary matrices and $\{\gamma_b\}$ is a boundary disturbance vector. Table 3.5.1 lists $[M_b]$, $[N_b]$, and $\{\gamma_b\}$ for the three boundary conditions given in Eqs. (3.3).

Solution by Distributed Transfer Functions

The solution of Eqs. (3.12) and (3.14) is given by

$$\eta(x) = -\frac{1}{\alpha_{ST}} \int_0^L [G(x,\xi)] f(\xi) d\xi \,\{e_2\} + [H(x)]\,\{\gamma_b\} \tag{3.15}$$

where the Green's function $[G(x,\xi)]$ and the boundary influence function $[H(x)]$ are

$$\begin{aligned} [G(x,\xi)] &= \begin{cases} [H(x)]\,[M_b]\,e^{-[F]\xi}, & x \ge \xi \\ -[H(x)]\,[N_b]\,e^{[F](L-\xi)}, & x \le \xi \end{cases} \\ [H(x)] &= e^{[F]x}\left([M_b] + [N_b]\,e^{[F]L}\right)^{-1} \end{aligned} \tag{3.16}$$

TABLE 3.5.1 Boundary Matrices and Boundary Disturbance Vector

Boundary Condition	*Left End (x = 0)*		*Right End (x = L)*	
	$[M_b]$	$\{\gamma_b\}$	$[N_b]$	$\{\gamma_b\}$
(B1) Fixed end	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{Bmatrix} w_0 \\ \bullet \end{Bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$	$\begin{Bmatrix} \bullet \\ w_L \end{Bmatrix}$
(B2) Free end	$\begin{bmatrix} 0 & -\alpha_{ST} \\ 0 & 0 \end{bmatrix}$	$\begin{Bmatrix} p_0 \\ \bullet \end{Bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & \alpha_{ST} \end{bmatrix}$	$\begin{Bmatrix} \bullet \\ p_L \end{Bmatrix}$
(B3) End spring	$\begin{bmatrix} k & -\alpha_{ST} \\ 0 & 0 \end{bmatrix}$	$\begin{Bmatrix} p_0 \\ \bullet \end{Bmatrix}$	$\begin{bmatrix} 0 & 0 \\ k & \alpha_{ST} \end{bmatrix}$	$\begin{Bmatrix} \bullet \\ p_L \end{Bmatrix}$

In the DTFM, these functions are called distributed transfer functions. The exponential matrix in Eqs. (3.16) is of the form

$$e^{[F]x} = \begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix} \tag{3.17}$$

By Eqs. (3.15) and (3.2), the static displacement and internal force of the system are expressed as

$$\begin{Bmatrix} w(x) \\ \sigma(x) \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \alpha_{ST} \end{bmatrix} \eta(x) = \begin{Bmatrix} w(x) \\ \sigma(x) \end{Bmatrix}_{ext} + \begin{Bmatrix} w(x) \\ \sigma(x) \end{Bmatrix}_{bd} \tag{3.18}$$

where

$$\begin{Bmatrix} w(x) \\ \sigma(x) \end{Bmatrix}_{ext} = -\frac{1}{\alpha_{ST}} \begin{bmatrix} 1 & 0 \\ 0 & \alpha_{ST} \end{bmatrix} \int_0^L [G(x,\xi)] f(\xi) d\xi \, \{e_2\} \tag{3.19}$$

which represents the response due to the external loads, and

$$\begin{Bmatrix} w(x) \\ \sigma(x) \end{Bmatrix}_{bd} = \begin{bmatrix} 1 & 0 \\ 0 & \alpha_{ST} \end{bmatrix} [H(x)] \{\gamma_b\} \tag{3.20}$$

which represents the response due to the boundary disturbances.

Given an external force $f(x)$, the integral in Eq. (3.19) can be obtained by exact quadrature. Thus, the solution given by Eq. (3.15) is exact and of closed form.

EXAMPLE 5.1

A longitudinal bar of stiffness EA and length L is fixed at its left end and connected to a spring of coefficient k at its right end. The bar is under a pointwise load f_0 applied at an interior point x_f, and experiences a boundary displacement u_0 at its left end. The bar is described by the equation

$$-EA\frac{d^2}{dx^2}u(x) = f_0\delta(x - x_f), \quad 0 \le x \le L$$

with the boundary conditions

$$\text{at } x = 0: \quad u(0) = u_0$$

$$\text{at } x = L: \quad ku(L) + EA\frac{d}{dx}u(L) = 0$$

The boundary matrices and boundary disturbance vector for the bar are

$$[M_b] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad [N_b] = \begin{bmatrix} 0 & 0 \\ k & EA \end{bmatrix}, \quad \{\gamma_b\} = \begin{pmatrix} u_0 \\ 0 \end{pmatrix}$$

The response of the bar, by Eq. (3.18), is obtained as

$$\begin{Bmatrix} w(x) \\ \sigma(x) \end{Bmatrix} = -f_0 \begin{bmatrix} \frac{1}{EA} & 0 \\ 0 & 1 \end{bmatrix} [G(x, x_f)] \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & EA \end{bmatrix} [H(x)] \begin{pmatrix} 1 \\ 0 \end{pmatrix} u_0$$

where

$$[G(x, x_f)] = \begin{cases} [H(x)] \begin{bmatrix} 1 & -x_f \\ 0 & 0 \end{bmatrix}, & x \geq x_f \\ -[H(x)] \begin{bmatrix} 0 & 0 \\ k & EA + k(L - x_f) \end{bmatrix}, & x \leq x_f \end{cases}$$

and

$$[H(x)] = \frac{1}{EA + kL} \begin{bmatrix} EA + k(L - x) & x \\ -k & 1 \end{bmatrix}.$$

Synthesis of Stepped Structures

The DTFM is convenient in synthesis and analysis of stepped bars and shafts as discussed in Section 3.4. As an example, consider the two-step shaft in Fig. 3.4.3. By Eq. (3.8), the torque balance at node 1 (the point where shafts 1 and 2 are interconnected) leads to

$$T_1(l_1) + k_c \theta_1(l_1) = T_2(0) \tag{3.21}$$

where $T_i(x)$ is the torque of the shaft i. By Eq. (3.15), the response of the shaft elements are

$$\text{for shaft 1:} \quad \eta_1(x) = \begin{pmatrix} \theta_1(x) \\ \theta_1'(x) \end{pmatrix} = -\frac{\tau_0}{GJ_1} [G_1(x, x_\tau)] \begin{pmatrix} 0 \\ 1 \end{pmatrix} + [H_1(x)] \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix} \tag{3.22a}$$

$$\text{for shaft 2:} \quad \eta_2(x) = \begin{pmatrix} \theta_2(x) \\ \theta_2'(x) \end{pmatrix} = -\frac{q}{GJ_2} \int_{l_2 - a}^{l_2} [G_2(x, \xi)] d\xi \begin{pmatrix} 0 \\ 1 \end{pmatrix} + [H_2(x)] \begin{pmatrix} \theta_1 \\ 0 \end{pmatrix} \tag{3.22b}$$

where $\theta_i' = d\theta_i/dx$, and θ_1 is the unknown rotational displacement at node 1. Write

$$[G_i(x, \xi)] = \begin{bmatrix} g_{11}^{(i)}(x, \xi) & g_{12}^{(i)}(x, \xi) \\ g_{21}^{(i)}(x, \xi) & g_{22}^{(i)}(x, \xi) \end{bmatrix}, \quad [H_i(x)] = \begin{bmatrix} h_{11}^{(i)}(x) & h_{12}^{(i)}(x) \\ h_{21}^{(i)}(x) & h_{22}^{(i)}(x) \end{bmatrix}, \quad i = 1, 2 \tag{3.23}$$

Substitute Eqs. (3.22) and (3.23) into the equilibrium equation (3.21), to obtain

$$\left\{ GJ_1 h_{22}^{(1)}(l_1) - GJ_2 h_{22}^{(2)}(0) + k_c \right\} \theta_1 = \tau_0 g_{22}^{(1)}(l_1, x_\tau) - GJ_1 h_{21}^{(1)}(l_1) \theta_0 - q \int_{l_2 - a}^{l_2} g_{22}^{(2)}(0, \xi) d\xi \tag{3.24}$$

Solving Eq. (3.24) for the nodal displacement θ_1 and substituting the result into Eqs. (3.22) yields the static response of the stepped shaft at every point. Again, the static response obtained is in exact and closed form.

The above distributed transfer function synthesis is applicable to stepped bar/shaft structures with more than two elements.

3.6 Quick Solution Guide

System Description

Type I: Single-Body Continua

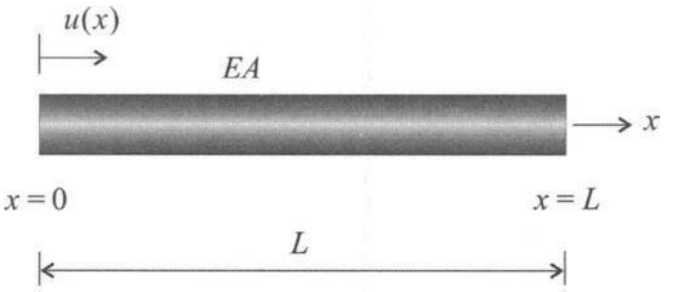
Elastic bar in longitudinal deformation

$$-EA\frac{d^2}{dx^2}u(x) = q(x)$$

$u(x)$: longitudinal displacement

$q(x)$: external load

EA : longitudinal rigidity



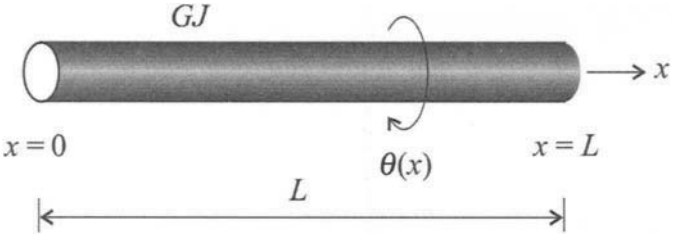
Circular shaft in torsional deformation

$$-GJ\frac{d^2}{dx^2}\theta(x) = \tau(x)$$

$\theta(x)$: rotation (twist)

$\tau(x)$: external torque

GJ : torsional rigidity



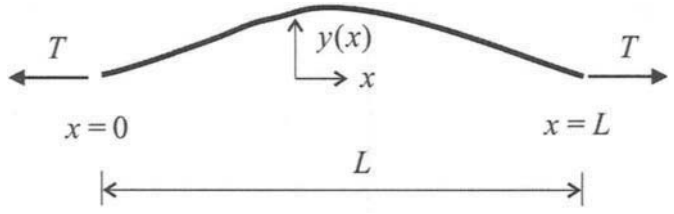
Taut string with lateral deflection

$$-T\frac{d^2}{dx^2}y(x) = p(x)$$

$y(x)$: lateral deflection

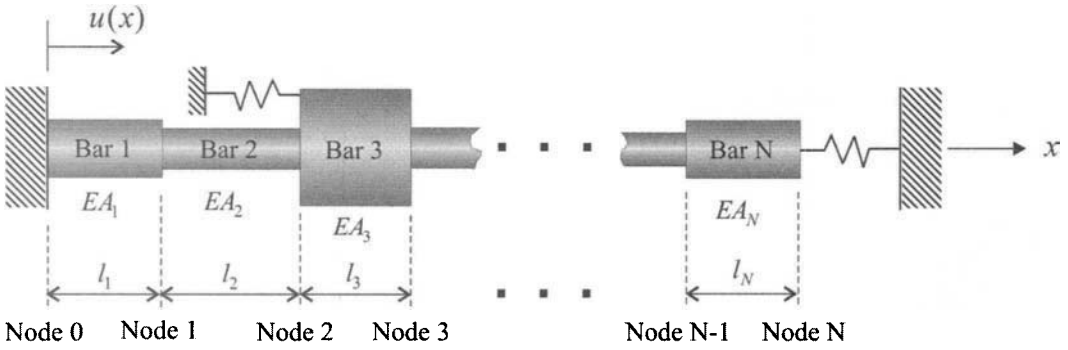
$p(x)$: lateral external load

T : string tension



Type II: Stepped Structures

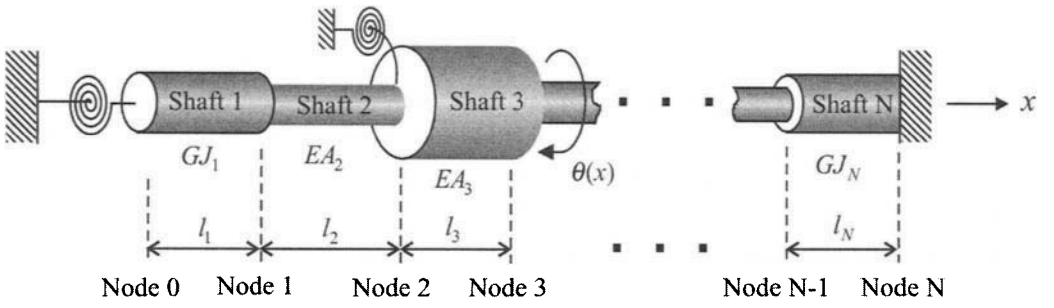
Stepped bar structure



Compatibility conditions at nodes

$$u_i(l_i) = u_{i+1}(0); \quad EA_i \frac{d}{dx} u_i(l_i) + k_c u_i(l_i) = EA_{i+1} \frac{d}{dx} u_{i+1}(0)$$

Stepped shaft structure



Compatibility conditions at nodes

$$\theta_i(l_i) = \theta_{i+1}(0); \quad GJ_i \frac{d}{dx} \theta_i(l_i) + k_c \theta_i(l_i) = GJ_{i+1} \frac{d}{dx} \theta_{i+1}(0)$$

Problem to Solve

Determination of static response (displacement and internal force) of a second-order system subject to external loads and boundary disturbances.

MATLAB Functions

This chapter contains a toolbox of MATLAB functions for static analysis of bars, shafts, and strings; see the table below. Refer to the corresponding window or section for the utility of each function. The application of some major functions is summarized in the following pages.

	System Setup	
`setbar`	Set up parameters and boundary conditions for a bar	Window 3.1
`setshaft`	Set up parameters and boundary conditions for a shaft	Window 3.1
`setstring`	Set up parameters and boundary conditions for a string	Window 3.1
	Static Analysis	
`force2os`	Compute system response subject to external loads	Window 3.2
`bd2os`	Compute system response due to boundary disturbances	Window 3.3
`plot2os`	Plot spatial distribution of system response and output computed response data	Window 3.4
	Stepped Bars and Shafts	
`setbar2`	Set up a stepped bar	Window 4.1
`setshaft2`	Set up a stepped shaft	Window 4.1
`step2osfb`	Compute the response of a stepped bar or stepped shaft subject to external loads and/or boundary disturbances	Window 4.2
`plot2os2`	Plot spatial distribution of the response of a stepped bar or shaft and output computed response data	Window 4.3

Utilities		
`TBdemo`	Show how the Toolbox works and what it can do	Section 3.1
`TBinfo`	Show the information of the Toolbox	Section 3.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 3.1
`systinfo` `systinfo2`	Display system parameters and boundary conditions	Sections 3.3.1 and 3.4.2

Solution Procedure

I) Static Analysis of Single-Body Continua

Step 1. Set up system parameter and boundary conditions.

For a longitudinal bar, type

```
>> setbar(EA, L, BC_Spec)
```

For a torsional shaft, type

```
>> setshaft(GJ, L, BC_Spec)
```

For a taut string, type

```
>> setstring(T, L, BC_Spec)
```

Step 2. Compute static response.

To compute response to external forces, type

```
>> y = force2os(Load_Spec);
```

To compute response subject to boundary disturbances, type

```
>> y = bd2os(BD_Spec);
```

To obtain total response, type:

```
>> y = force2os(Load_Spec) + bd2os(BD_Spec);
```

Step 3. Plot the computed response.

```
>> plot2os(y)
```

where y is a matrix of response data that is obtained by functions `force2os` and `bd2os`

Note 1. The `setbar`, `setshaft`, `setstring`, `force2os`, `bd2os`, and `bd2os` are MATLAB functions from the Toolbox; see Windows 3.1 to 3.4 for details.

Note 2. The `BC_Spec`, `Load_Spec`, and `BD_Spec` are matrices specifying boundary conditions, external forces, and boundary disturbances, respectively. Specification of these matrices is given as follows.

(i) Specification of boundary conditions and boundary disturbances

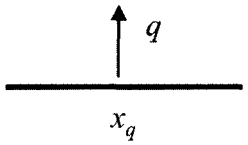
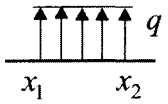
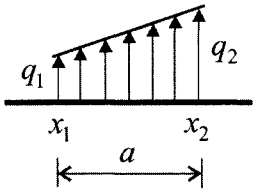
Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`
Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

Boundary Conditions	*Left Boundary (x = 0)*		*Right Boundary (x = L)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B1) Fixed end	[1]	$[w_0]$	[1]	$[w_L]$
(B2) Free end	[2]	$[p_0]$	[2]	$[p_L]$
(B3) End spring	[3 *k*]	$[p_0]$	[3 *k*]	$[p_L]$

(ii) Specification of external loads

$$\texttt{Load\_Spec} = \begin{bmatrix} \texttt{Load\_1} \\ \texttt{Load\_2} \\ \vdots \\ \texttt{Load\_Nf} \end{bmatrix}$$

where `Nf` is the number of loads. Each row of the matrix specifies an external load according to the following table.

External Load		*Load Specification Vector*
(L1) Pointwise force 	$f(x) = q\delta(x - x_q)$	`Load_j = [0` x_q `q 0 0]`
(D1) Uniform distribution 	$f(x) = q$ for $x_1 \le x \le x_2$	`Load_j = [1` x_1 x_2 q `0]`
(D2) Linear distribution 	$f(x) = q_1 + \frac{q_2 - q_1}{a}(x - x_1)$ for $x_1 \le x \le x_2$	`Load_j = [2` x_1 x_2 q_1 q_2`]`

II) Static Analysis of Stepped Bars and Shafts

Step 1. Set up system parameter, boundary conditions, and constraints.

For a stepped bar, type

```
>> setbar2(Element_Spec, BC_Spec, Constr_Spec)
```

For a torsional shaft, type

```
>> setshaft2(Element_Spec, BC_Spec, Constr_Spec)
```

Step 2. Compute and display static response.

Type

```
>> step2osfb(Load_Spec, BD_Spec)
```

The `setbar2`, `setshaft2`, and `step2osfb` are MATLAB functions from the Toolbox; see Windows 4.1 and 4.2 for details.

The Element_Spec, BC_Spec, Constr_Spec, Load_Spec, and BD_Spec are matrices specifying elements, boundary conditions, spring constraints, external forces, and boundary disturbances, respectively. Specification of these matrices is given below.

(i) Specification of elements of a stepped structure
For a stepped bar:

$$\texttt{Element\_Spec} = \begin{bmatrix} EA_1 & l_1 \\ EA_2 & l_2 \\ \vdots & \vdots \\ EA_N & l_N \end{bmatrix}$$

For a stepped shaft:

$$\texttt{Element\_Spec} = \begin{bmatrix} GJ_1 & l_1 \\ GJ_2 & l_2 \\ \vdots & \vdots \\ GJ_N & l_N \end{bmatrix}$$

Here, the ith row of the matrix contains the stiffness and length of the ith element.

(ii) Specification of boundary conditions
BC_Spec is the same as that for a single-body continuum.

(iii) Specification of spring constraints at interior nodes $(1, 2, \ldots, N-1)$

$$\texttt{Constr\_Spec} = \begin{bmatrix} \texttt{Constr\_1} \\ \texttt{Constr\_2} \\ \vdots \\ \texttt{Constr\_Nc} \end{bmatrix}$$

The jth row of the matrix specifies the jth spring constraint by

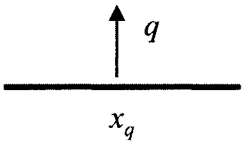
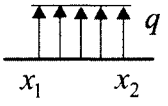
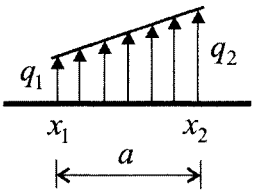
$$\texttt{Constr\_j} = [\texttt{node\_no} \quad \texttt{kc}]$$

where node_no is the node number, and kc is the spring coefficient. If a stepped structure has no constraints, Constr_Spec = 0 or [ ].

(iv) Specification of external loads

$$\texttt{Load\_Spec} = \begin{bmatrix} \texttt{Load\_1} \\ \texttt{Load\_2} \\ \vdots \\ \texttt{Load\_Nf} \end{bmatrix}$$

where Nf is the number of loads. Each row of the matrix specifies an external load according to the following table.

Load	*Load Specification Vector*
(L1) Pointwise force  $f(x) = q\delta(x - x_q)$	`Load_j = [element_no 0` x_q `q 0 0]`
(D1) Uniform distribution  $f(x) = q$ for $x_1 \leq x \leq x_2$	`Load_j = [element_no 1` x_1 x_2 `q 0]`
(D2) Linear distribution  $f(x) = q_1 + \frac{q_2 - q_1}{a}(x - x_1)$ for $x_1 \leq x \leq x_2$	`Load_j = [element_no 2` x_1 x_2 q_1 q_2`]`

(v) Specification of boundary disturbances
`BD_Spec` is the same as that for a single-body continuum.

3.7 References

1. Bedford A, Liechti K M 2000 *Mechanics of Materials*, Prentice-Hall: Upper Saddle River, New Jersey.
2. Gere J M, Timoshenko, S P 1997 *Mechanics of Materials*, 4th ed., PWS Publishing Co.: Boston.
3. Popov E P 1998 *Engineering Mechanics of Solids*, 2nd ed., Prentice-Hall: Upper Saddle River, New Jersey.
4. Riley W F, Sturges L D, Morris D H 1999 *Mechanics of Materials*, 5th ed., John Wiley & Sons, Inc.: New York.
5. Shames I H, Pitarresi J M 2000 *Introduction to Solid Mechanics*, 3rd ed., Prentice-Hall: Upper Saddle River, New Jersey.

This Page Intentionally Left Blank

4 Buckling Analysis of Columns

Inside
- Getting Started
- Uniform Columns
- Stepped and Nonuniform Columns
- The Distributed Transfer Function Method
- Quick Solution Guide
- References

4.1 Getting Started

What Is in This Chapter

This Toolbox is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for buckling analysis of columns described by the Euler-Bernoulli beam model.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on the computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

Quick Tutorial

To show how to use the Toolbox, consider a column of bending stiffness EI and length L, which is clamped at the left end and supported by a spring of coefficient k_t at the right end (see Fig. 4.1.1). Let the parameters of the column be

$$EI = 25, \quad L = 2, \quad k_t = 4.75$$

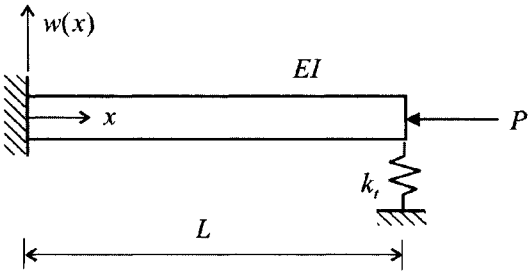


FIGURE 4.1.1 A column under an axial load.

To determine the buckling loads and mode shapes of the column, in the MATLAB command window, type the following:

```
>> EI = 25; L = 2; kt = 4.75;
>> BC_Spec = [2 5 0 kt];
>> setcolumn(EI, L, BC_Spec)
```

where >> is the prompt in the MATLAB command window. This yields the first four buckling loads

```
P1 = 23.0559
P2 = 139.7018
P3 = 385.8469
P4 = 755.8007
```

and the associate mode shapes plotted in Fig. 4.1.2.

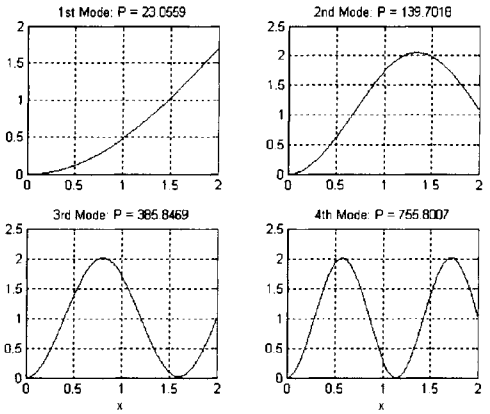


FIGURE 4.1.2 Buckling mode shapes.

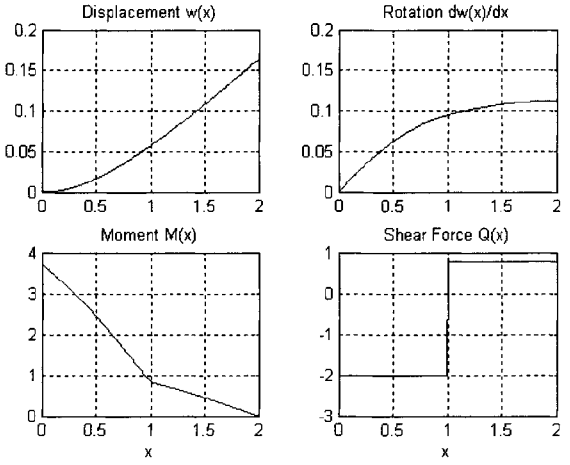


FIGURE 4.1.3 Beam-column response distribution.

Assume that the same member is subject to an axial load $P = 15$ and a pointwise transverse load $f_0 = 2.8$ at $x = 1.0$. The spatial distribution of the deflection and internal forces of the member, which is a beam-column now, can be plotted by

```
>> load = [0 1 2.8]; p = 15;
>> beamcolumn(load, p)
```

which yields Fig. 4.1.3.

In the above buckling analysis, two functions from the Toolbox have been called:

(i) Function **setcolumn** that computes and displays the buckling loads and mode shapes of the column.
(ii) Function **beamcolumn** that computes the response of the beam-column subject to a transverse load.

These functions, along with others, will be introduced in the subsequent sections.

For a quick solution, the reader may refer directly to the Quick Solution Guide (Section 4.5), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To better understand how the toolbox works, the reader is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 4.6.

4.2 Uniform Columns

Long slender members under axial compressive loadings are called *columns*. When the axial force is large enough, dramatic lateral deflection of a column occurs, which is often referred to as *buckling*. Buckling of a column may lead to sudden failure of structures, and thus is of special concern in design of structures and machines. In this section, buckling of uniform columns is considered. Buckling of nonuniform columns is addressed in Section 4.3.

4.2.1 Column Buckling Theory

In Fig. 4.2.1, a straight uniform column is under an axial compression load P at its two ends. Assume that the compression load increases from zero. When P is small, the column remains

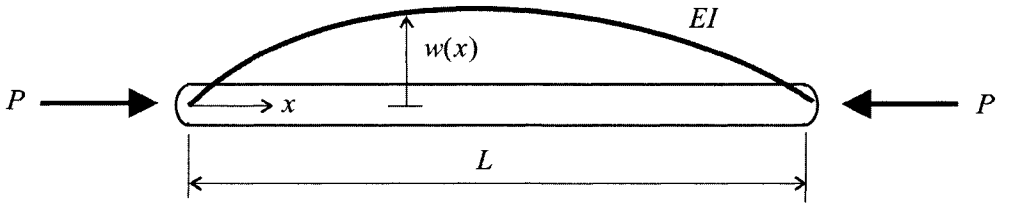


FIGURE 4.2.1 A column subject to an axial compression load.

straight; that is, the transverse displacement of the column is zero. As P reaches a critical value, the column suddenly buckles sideways. This critical value is called *buckling load,* and the corresponding displacement is called *buckling mode shape.* The least compression load (P_{cr}) at which the column buckles is called the *Euler buckling load.*

The transverse buckling deflection $w(x)$ of a long uniform column can be described by the following Euler-Bernoulli beam equation (Reference 3)

$$EI\frac{d^4}{dx^4}w(x) + P\frac{d^2}{dx^2}w(x) = 0, \quad 0 \le x \le L \tag{4.1}$$

where P is the axial compression load, and EI and L are the bending stiffness and length of the column, respectively. The bending moment $M(x)$ and shear force $Q(x)$ of the column are

$$\begin{aligned} M(x) &= EI\frac{d^2}{dx^2}w(x) \\ Q(x) &= EI\frac{d^3}{dx^3}w(x) + P\frac{d}{dx}w(x) \end{aligned} \tag{4.2}$$

Seven types of boundary conditions of columns are given in Table 4.2.1. Equation (4.1), along with specific boundary conditions, forms an eigenvalue problem, for which the buckling load P is an eigenvalue and the buckling mode shape $w(x)$ is the associate eigenfunction.

To solve the buckling (eigenvalue) problem, rewrite Eq. (4.1) as

$$\frac{d^4}{dx^4}w(x) + \beta^2\frac{d^2}{dx^2}w(x) = 0; \quad \beta^2 = \frac{P}{EI} \tag{4.3}$$

The solution of Eq. (4.3) is of the form

$$w(x) = a_1 + a_2x + a_3\cos\beta x + a_4\sin\beta x \tag{4.4}$$

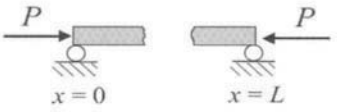
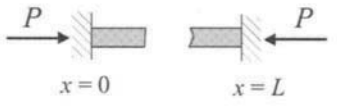
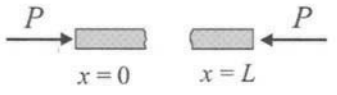
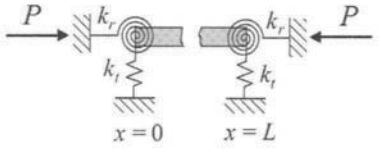
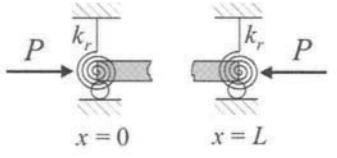
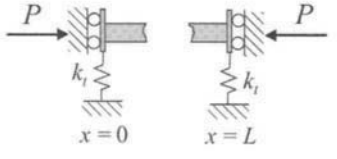
Substituting Eq. (4.4) into the boundary conditions of the column leads to a set of algebraic equations

$$[\Phi(\mu)]\{A\} = 0 \tag{4.5}$$

where $\{A\} = (a_1 \quad a_2 \quad a_3 \quad a_4)^T$, $[\Phi(\mu)]$ is a 4-by-4 matrix whose elements are functions of the parameter

$$\mu = \beta L = \sqrt{\frac{P}{EI}}L \tag{4.6}$$

TABLE 4.2.1 Boundary Conditions of a Column
(k_r - torsional spring coefficient; k_t - translational spring coefficient)

Boundary Type	*Boundary Conditions*
(B1) Pinned or hinged end 	Left end: $w(0) = 0, \quad \frac{d^2w(0)}{dx^2} = 0$ Right end: $w(L) = 0, \quad \frac{d^2w(L)}{dx^2} = 0$
(B2) Clamped or fixed end 	Left end: $w(0) = 0, \quad \frac{dw(0)}{dx} = 0$ Right end: $w(L) = 0, \quad \frac{dw(L,t)}{dx} = 0$
(B3) Free end 	Left end: $\frac{d^2w(0)}{dx^2} = 0, \quad EI\frac{d^3w(0)}{dx^3} + P\frac{dw(0)}{dx} = 0$ Right end: $\frac{d^2w(L)}{dx^2} = 0, \quad EI\frac{d^3w(L)}{dx^3} + P\frac{dw(L)}{dx} = 0$
(B4) Sliding or guided end	Left end: $\frac{dw(0)}{dx} = 0, \quad \frac{d^3w(0)}{\partial x^3} = 0$ Right end: $\frac{dw(L,t)}{dx} = 0, \quad \frac{d^3w(L)}{dx^3} = 0$
(B5) Elastic-support 	Left end: $k_r\frac{dw(0)}{dx} - EI\frac{d^2w(0)}{dx^2} = 0, \quad k_t w(0) + EI\frac{d^3w(0)}{dx^3} + P\frac{dw(0)}{dx} = 0$ Right end: $k_r\frac{dw(L)}{dx} + EI\frac{d^2w(L)}{dx^2} = 0, \quad k_t w(L) - EI\frac{d^3w(L)}{dx^3} - P\frac{dw(L)}{dx} = 0$
(B6) Hinged-elastic end 	Left end: $w(0) = 0, \quad k_r\frac{dw(0)}{dx} - EI\frac{d^2w(0)}{dx^2} = 0$ Right end: $w(L) = 0, \quad k_r\frac{dw(L)}{dx} + EI\frac{d^2w(L)}{dx^2} = 0$
(B7) Sliding-elastic end 	Left end: $\frac{dw(0)}{dx} = 0, \quad k_t w(0) + EI\frac{d^3w(0)}{dx^3} + P\frac{dw(0)}{dx} = 0$ Right end: $\frac{dw(L)}{dx} = \theta_L, \quad k_t w(L) - EI\frac{d^3w(L)}{dx^3} - P\frac{dw(L)}{dx} = 0$

The characteristic equation of the column is

$$\det[\Phi(\mu)] = 0 \tag{4.7}$$

the roots of which, $\mu_1, \mu_2, \ldots$, are related to the buckling loads by

$$P_k = \mu_k^2 \frac{EI}{L^2}, \quad k = 1, 2, \ldots \tag{4.8}$$

With μ_k, the associate buckling mode shapes can be derived from Eqs. (4.4) and (4.5).

Let the roots of Eq. (4.7) be arranged in an ascending order, namely, $\mu_1 < \mu_2 < \cdots$. The Euler buckling load then is $P_{cr} = P_1 = \mu_1^2 EI/L^2$. For columns with different boundary conditions, the Euler buckling load can be written as

$$P_{cr} = \alpha \frac{\pi^2 EI}{L^2} = \frac{\pi^2 EI}{(\kappa L)^2} \tag{4.9}$$

where α is the end condition constant and κ the effective length factor. These parameters are listed in Table 4.2.2, for several column configurations. For columns of general configurations as described in Table 4.2.1, Euler buckling loads and associate mode shapes can be conveniently determined by the Toolbox of this chapter; see Section 4.2.2.

As an example, consider a column with clamped-pinned end conditions (B2-B1):

$$w(0) = 0, \quad \frac{d}{dx} w(0) = 0; \quad w(L) = 0, \quad \frac{d^2}{dx^2} w(L) = 0 \tag{4.10}$$

Substituting Eq. (4.4) into Eqs. (4.10) gives

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & \beta \\ 1 & L & \cos\beta L & \sin\beta L \\ 0 & 0 & \cos\beta L & \sin\beta L \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = 0 \tag{4.11}$$

from which, transcendental characteristic equation is obtained according to Eq. (4.7):

$$\mu - \tan\mu = 0 \tag{4.12}$$

with $\mu = \beta L$. The eigenfunction associated with the kth eigenvalue by Eq. (4.11) is

$$w_k(x) = A_k \left\{ 1 - \frac{x}{L} - \cos\left(\mu_k \frac{x}{L}\right) + \frac{1}{\mu_k} \sin\left(\mu_k \frac{x}{L}\right) \right\} \tag{4.13}$$

TABLE 4.2.2 End Condition Constant and Effective Length Factor

End Conditions	*End Condition Constant, α*	*Effective Length Factor, κ*
Clamped-Free	1/4	2
Pinned-Sliding	1/4	2
Pinned-Pinned	1	1
Clamped-Sliding	1	1
Clamped-Pinned	2	0.7
Clamped-Clamped	4	1/2

where A_k is a nonzero constant. The first four roots of Eq. (4.12) are found to be

$$\mu_1 = 4.4934, \quad \mu_2 = 7.7253, \quad \mu_3 = 10.9041, \quad \mu_4 = 14.0662$$

Thus, the Euler buckling load of the column is

$$P_{cr} = \mu_1^2 \frac{EI}{L^2} = 2.05 \frac{\pi^2 EI}{L^2}$$

and the associate mode shape is

$$w_1(x) = A_1 \left\{ 1 - \frac{x}{L} - \cos\left(4.493 \frac{x}{L}\right) + 0.2225 \sin\left(4.493 \frac{x}{L}\right) \right\}.$$

4.2.2 Solution by the Toolbox

The Toolbox of this chapter provides a MATLAB function `setcolumn` for buckling analysis of columns with various end conditions listed in Table 4.2.1; see Window 2.1. The solution technique used by function `setcolumn` is described in Section 4.4.

Window 2.1. Function `setcolumn`

MATLAB Function: **`setcolumn`**

Purpose:
To set up a column and to compute its buckling loads and associate mode shapes.

Synopsis:

```
setcolumn(EI, L, BC_Spec)
setcolumn(EI, L, BC_Spec, n_mode)
```

Description:
`setcolumn(EI, L, BC_Spec)` undertakes the following three tasks:

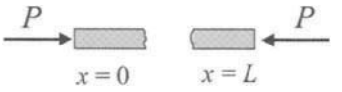
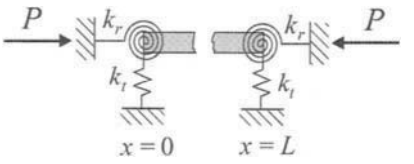
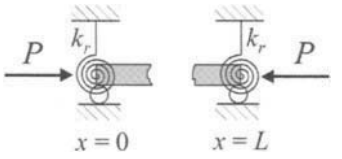
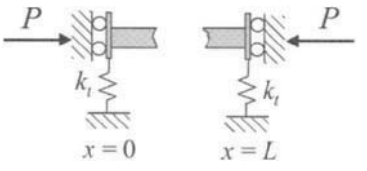
(a) It inputs the bending stiffness `EI` and length `L` of the column.
(b) It specifies the boundary conditions of the column by a row vector `BC_Spec`, whose formation is given in Table 4.2.3.
(c) It computes the buckling loads and associate mode shapes of the column.

`setcolumn(EI, L, BC_Spec, n_mode)` selects the first `n_mode` buckling modes for numerical analysis. By default, `n_mode` = 4.

Note 1. Function `setcolumn` displays computed eigenvalues by three parameters: P, buckling loads; $\beta_k = \sqrt{P/EI}$; and $\mu_k = \beta_k L$. The function also displays buckling mode shapes in terms of Eq. (4.4). While four buckling modes are displayed by default, the Euler buckling load is of the primary interest in practice.

Note 2. Function `setcolumn` must be executed before other functions for buckling analysis of uniform columns can be called. However, for the same column in different situations as discussed in Sections 4.2.3 to 4.2.5, `setcolumn` only needs to be executed once.

TABLE 4.2.3 Specification of Boundary Conditions

Boundary Condition *(k_r - torsional spring coefficient; k_t - translational spring coefficient)*	**`BC_Spec = [BC_Left BC_Right]`**	
	Left Boundary (x = 0) `BC_Left`	*Right Boundary (x = L)* `BC_Right`
(B1) Pinned or hinged end	[1]	[1]
(B2) Clamped or fixed end	[2]	[2]
(B3) Free end 	[3]	[3]
(B4) Sliding or guided end	[4]	[4]
(B5) Elastic-support 	[5 k_r k_t]	[5 k_r k_t]
(B6) Hinged-elastic end 	[6 k_r]	[6 k_r]
(B7) Sliding-elastic end 	[7 k_t]	[7 k_t]

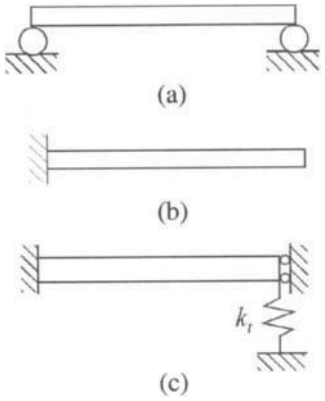


FIGURE 4.2.2

The boundary specification by Table 4.2.3 is illustrated on three column configurations shown in Fig. 4.2.2:

(a) Pinned-pinned ends (B1-B1)

```
BC_Spec = [1  1]
```

(b) Clamped-free ends (B2-B3)

```
BC_Spec = [2  3]
```

(c) Clamped and Sliding-elastic ends (B2-B7)

```
BC_Spec = [1  7  kt]
```

EXAMPLE 2.1

Consider a column with parameters $EI = 12$ and $L = 2$ and pinned-sliding ends. Executing the command

```
>> setcolumn(12, 2, [1 4])
```

yields the first four buckling loads and the associate mode shapes:

```
First 4 Buckling Loads
----------------------
P = buckling load
beta = sqrt(P/EI)
mu = beta*L, nondimensional eigenvalue

      P        beta         mu
   7.4022     0.7854      1.5708
  66.6198     2.3562      4.7124
 185.0551     3.9270      7.8540
 362.7080     5.4978     10.9956
```

```
Buckling Mode Shapes
--------------------
 w(x)=a1+a2*x+a3*cos(beta*x)+a4*sin(beta*x)
Mode No.    beta          a1          a2          a3          a4
1.0000     0.7854          0           0           0      1.0000
2.0000     2.3562          0           0           0      1.0000
3.0000     3.9270          0           0           0      1.0000
4.0000     5.4978          0           0           0      1.0000
```

Also, function `setcolumn` plots the mode shapes in Fig. 4.2.3.

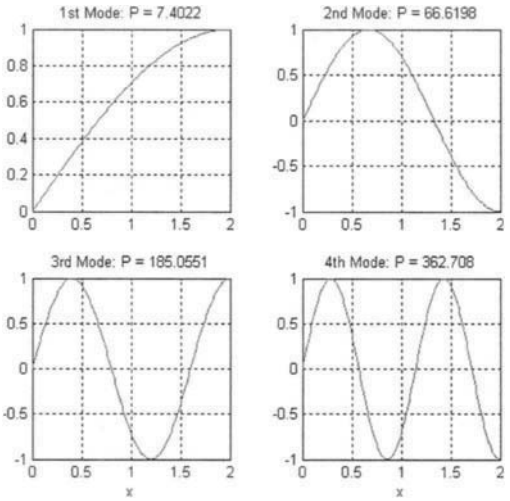


FIGURE 4.2.3

To get access to the information on system parameters, end conditions, and eigensolutions, function **`systinfo`** from the Toolbox can be called at any time.

Besides `setcolumn`, functions `eigcolumn` and `plotmsh` from the Toolbox are also useful for buckling analysis of columns; see Windows 2.2 and 2.3.

Window 2.2. Function `eigcolumn`

MATLAB Function: **`eigcolumn`**

Purpose:
To determine the buckling loads and mode shapes of a column.

Synopsis:

```
eigcolumn(n)
[y, msh] = eigcolumn(n)
```

Window 2.2. Function `eigcolumn` (*Continued*)

Description:
`eigcolumn(n)` computes and displays the first `n` buckling loads and mode shapes.

`[y, msh] = eigcolumn(n)` returns the first `n` buckling loads in a vector `y` and the coefficients of associate buckling mode shapes in an n-by-five matrix `msh`. The jth row of the matrix is of the form

`msh(j,:) =` $[\beta \quad a_1 \quad a_2 \quad a_3 \quad a_4]$

which contains the coefficients of the jth mode shape given in Eq. (4.4).

Window 2.3. Function `plotmsh`

MATLAB Function: **`plotmsh`**

Purpose:
To plot a buckling mode shape of a column.

Synopsis:

```
plotmsh(k)
plotmsh(k, npts)
[y, x] = plotmsh(k, npts)
```

Description:
`plotmsh(k)` plots the kth buckling mode shape over the spatial region $[0, L]$.

`plotmsh(k, npts)` plots the kth buckling mode shape at `npts` equally-spaced points in the region $[0, L]$. By default, `npts` = 201.

`[y, x] = plotmsh(k, npts)` returns the kth mode shape in a vector `y` and spatial points of computation in a vector `x`. With `x` and `y`, the buckling mode shape can be plotted by the MATLAB function `plot(x, y)`.

EXAMPLE 2.2

For the column in Example 2.1, the 1st and 3rd mode shapes are plotted in Fig. 4.2.4 by

```
>> setcolumn(12, 2, [1 4], 8)
>> [y1, x] = plotmsh(1);
>> y2 = plotmsh(3);
>> clf
>> plot(x, y1, x, y2)
>> grid
>> xlabel('x')
>> ylabel('w(x)')
```

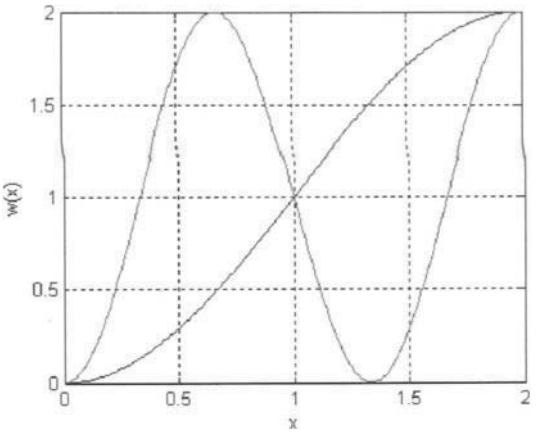


FIGURE 4.2.4

4.2.3 Eccentric Loading

In reality, the axial load P is always applied to the column at a short distance (eccentricity) e from the centroid of the cross-section. Because of this, a column never suddenly buckles, but bends immediately after the load is applied and deflects more as the load increases. A typical eccentrically loaded column is shown in Fig. 4.2.5(a). The eccentric loading is statically equivalent to an axial load P applied to the centroid of the cross-section and a moment Pe; see Fig. 4.2.5(b). Under such loading, the boundary conditions of the column become

$$w(0) = 0, \quad EI\frac{d^2}{dx^2}w(0) = -Pe$$

$$EI\frac{d^2}{dx^2}w(L) = -Pe, \quad EI\frac{d^3}{dx^3}w(L) + P\frac{d}{dx}w(L) - k_t w(L) = 0$$

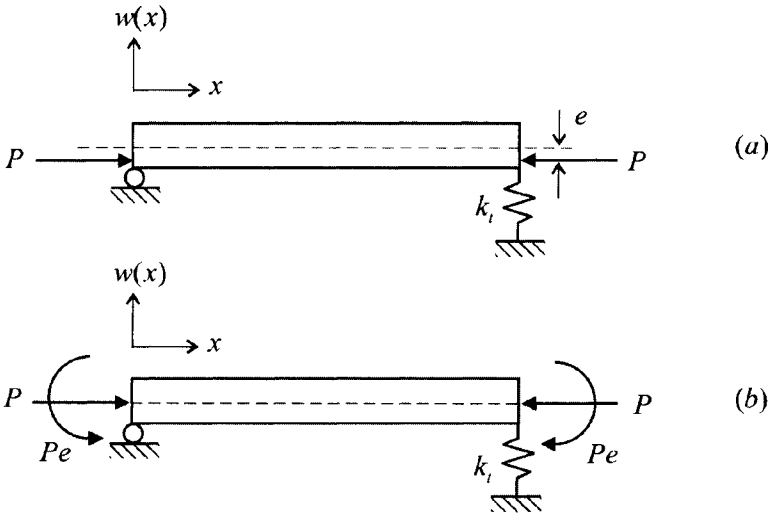


FIGURE 4.2.5 An eccentrically loaded column.

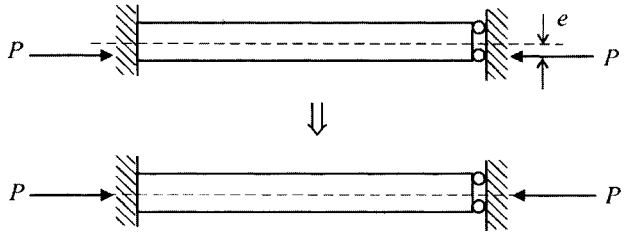


FIGURE 4.2.6

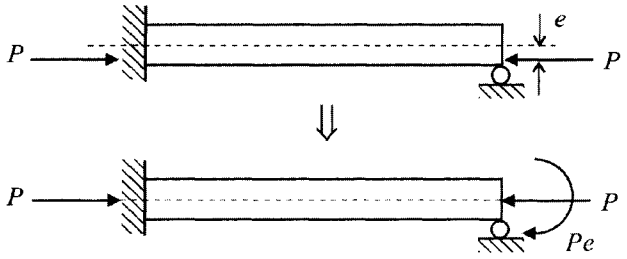


FIGURE 4.2.7

Thus, by Eq. (4.1) and the above boundary conditions, $w(x)$ is a function of the end moment Pe. In other words, with an eccentricity e, the column will undergo lateral deflection even if the axial load is very small. It can be shown that the deflection becomes infinity as the axial load approaches the Euler buckling load P_{cr}.

It should be pointed out that, under an eccentric load, boundaries B2 (clamped), B4 (sliding), and B7 (sliding-elastic) cannot produce end moment Pe because those ends cannot rotate; see Fig. 4.2.6, for example. So, if both ends of a column are among types B2, B4, and B7, no effect of eccentric loading shall appear. On the other hand, if only one end of the column is among types B2, B4, and B7, eccentric loading will generate a moment Pe at the other end; see Fig. 4.2.7 for instance. The column will deflect even if the axial force P is less than the Euler buckling load.

The Toolbox of this chapter provides functions `eccenload` and `eccenload1` for plotting the deflection of eccentrically loaded columns; see Windows 2.4 and 2.5.

Window 2.4. Function eccenload

MATLAB Function: **eccenload**

Purpose:
To plot the response of an eccentrically loaded column versus the axial load.

Synopsis:

```
eccenload(e, xr, pf, npts)
[y, p] = eccenload(e, xr, pf, npts)
```

Window 2.4. Function eccenload (*Continued*)

Description:
`eccenload(e, xr, pf, npts)` plots the response of the eccentrically loaded column at point `xr`, versus the axial load. Here, `e` is the eccentricity of the axial load, `pf` is a parameter defining a range of the axial load P by $0 \leq P \leq$ `pf`, and `npts` is the number of equally-spaced points in the range `[0, pf]` for computation. The `pf` must be less than the Euler buckling load P_{cr} of the column. By default, `xr` = $L/2$ where L is the column length, `pf` = $0.99P_{cr}$ and `npts` = 100.

`[y, p] = eccenload(e, xr, pf, npts)` returns the computed column response at `xr` in a four-row matrix `y` and the values of the axial load in a vector `p`. Here, `y(1,:)` stores the column deflection, `y(2,:)` the rotation, `y(3,:)` the bending moment, and `y(4,:)` the shear force. With `y` and `p`, the load-column response curves can be plotted by the MATLAB function `plot`. For example, for the load-bending moment curve, use `plot(p, y(3,:))`.

Window 2.5. Function eccenload1

MATLAB Function: **eccenload1**

Purpose:
To plot the spatial distribution of the response of an eccentrically loaded column.

Synopsis:

```
eccenload1(e, p, npts)
[y, x] = eccenload1(e, p, npts)
```

Description:
`eccenload1(e, p, npts)` plots the spatial distribution of the response of the column that is under an axial load `p` with eccentricity `e`. Here, `npts` is the number of equally-spaced points in the spatial domain [0, L], with L being the column length. The value of `p` must be less than the Euler buckling load P_{cr} of the column. By default, `p` = $0.5P_{cr}$ and `npts` = 201.

`[y, x] = eccenload1(e, p, npts)` returns the computed column response in a four-row matrix `y` and the values of the spatial points in the domain [0, L] in a vector `x`. Here, `y(1,:)` stores the spatial distribution of the column deflection, `y(2,:)` the rotation, `y(3,:)` the bending moment, and `y(4,:)` the shear force. With `y` and `x`, the response of the column can be plotted by the MATLAB function `plot`. For instance, for the spatial distribution of shear force, use `plot(x, y(4,:))`.

EXAMPLE 2.3

Consider the eccentrically loaded column in Fig. 4.2.5, with parameters $EI = 20$, $L = 2$, and $k_t = 100$. The load-deflection curves of the column at $x = 1.4$ with eccentricity

$e = 0.05, 0.1$ and 0.15 are plotted in Fig. 4.2.8 by

```
>> setcolumn(20, 2, [1 5 0 100])
>> [y1, p]=eccenload(0.05, 1.4);
>> y2=eccenload(0.1, 1.4);
>> y3=eccenload(0.15, 1.4);
>> clf
>> plot(y1(1,:), p, y2(1,:), p, y3(1,:), p)
>> grid
>> title('Load-Deflection Curves ... for e = 0.05, 0.1 and 0.15')
>> xlabel('w at x=1.4')
>> ylabel('P')
```

Here the Euler buckling load of the column is $P_{cr} = 49.348$, which is obtained when function `setcolumn` is called.

Next, consider the spatial distribution of the column deflection. Let the eccentricity e be 0.05. For the axial load $P = 10$, 20, and 30, the column deflection is plotted in Fig. 4.2.9 by the commands:

```
>> [y1, x]=eccenload1(0.05, 10);
>> y2=eccenload1(0.05, 20);
>> y3=eccenload1(0.05, 30);
>> clf
>> plot(x, y1(1, :), x, y2(1,:), x, y3(1,:))
>> grid
>> title('Deflection at e = 0.05, for P = 10, 20 and 30')
>> xlabel('x')
>> ylabel('W(x)')
```

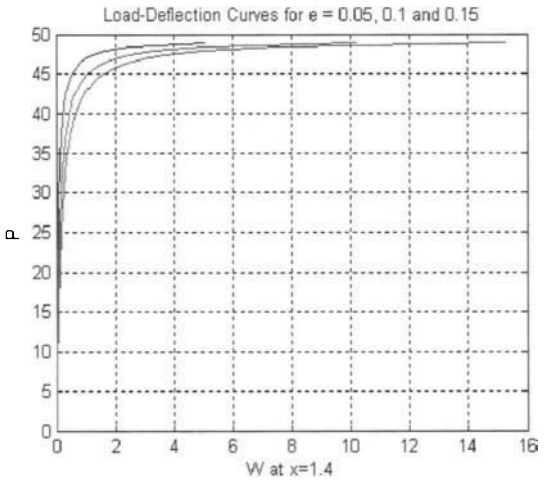


FIGURE 4.2.8

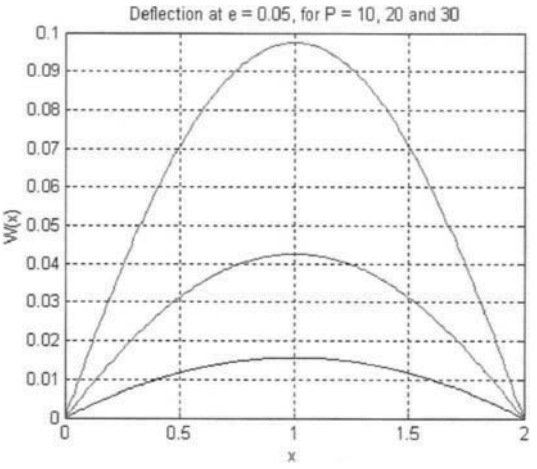


FIGURE 4.2.9

4.2.4 Beam-Column Problem

The beam-column is a beam that carries transverse loading, and at the same time, is subject to an axial load (see Fig. 4.2.10). The transverse deflection $w(x)$ of a uniform Euler-Bernoulli beam-column, under a transverse force $f(x)$ and an axial load P, is governed by

$$EI\frac{d^4}{dx^4}w(x) + P\frac{d^2}{dx^2}w(x) = f(x), \quad 0 \le x \le L \tag{4.14}$$

where EI and L are the bending stiffness and length of the beam-column, respectively. The boundary conditions of the beam-column are given in Table 4.2.1. Also, the bending moment $M(x)$ and shear force $Q(x)$ of the beam-column are the same as given in Eq. (4.2).

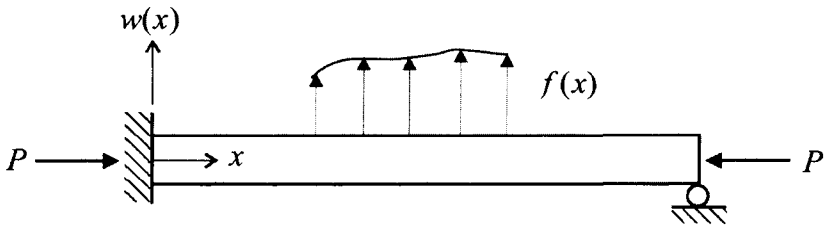


FIGURE 4.2.10 A beam-column under a transverse load.

The Toolbox of this chapter provides a function `beamcolumn` for determination of the response of a beam-column; see Window 2.6.

Window 2.6. Function `beamcolumn`

MATLAB Function: **`beamcolumn`**

Purpose:
To plot the response of a beam-column under a transverse force.

Window 2.6. Function `beamcolumn` (*Continued*)

Synopsis:

```
beamcolumn(Load_Spec, p, npts)
[y, x] = beamcolumn(Load_Spec, p, npts)
```

Description:

`beamcolumn(Load_Spec, p, npts)` plots the spatial distribution of the response of a beam-column, subject to an axial load `p` and an external load described by vector `Load_Spec`. Here `npts` is the number of equally-spaced points in the domain [0, L] for computation, with L being the length of the beam-column. Specification of vector `Load_Spec` is given in Table 4.2.4. By default, `p` is equal to the half of the Euler buckling load of the corresponding column, and `npts` = 201.

`[y, x] = beamcolumn(Load_Spec, p, npts)` returns the beam-column response in a four-row matrix `y` and the spatial points of computation in vector `x`. Here, `y(1,:)` stores the displacement $w(x)$, `y(2,:)` the rotation $dw(x)/dx$, `y(3,:)` the bending moment $M(x)$, and `y(4,:)` the shear force $Q(x)$. With `y` and `x`, the spatial distribution of the beam-column response can be plotted by the MATLAB function `plot`. For example, for the deflection $w(x)$, use `plot(x, y(1,:))`; and for the shear force $Q(x)$, use `plot(x, y(4,:))`.

As an example, for the load of parabolic distribution shown in Fig. 4.2.11,

$$f(x) = q\frac{(x - x_1)^2}{a^2}, \quad x_1 \le x \le x_2$$

the load specification vector is

$$\texttt{Load\_Spec} = [3 \;\; x_1 \;\; x_2 \;\; c_0 \;\; c_1 \;\; c_2]$$

where $c_0 = \frac{q}{a^2}x_1^2$, $c_1 = -\frac{2q}{a^2}x_1$, $c_2 = \frac{q}{a^2}$.

If a beam-column is subject to N transverse loads specified by vectors `Load1`, `Load2`,..., `LoadN`, its total response can be computed by superposition:

```
[y1, x] = beamcolumn(Load1, p);
y_total = y1 + beamcolumn(Load2, p) + ··· + beamcolumn(LoadN, p);
```

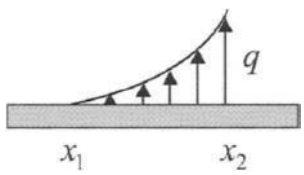
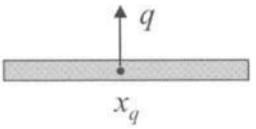


FIGURE 4.2.11 A load of parabolic distribution.

TABLE 4.2.4 Specification of External Loads

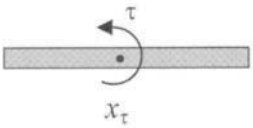
(L1) Pointwise force



$f(x) = q\delta(x - x_q)$

Load_Spec = [0 x_q q]

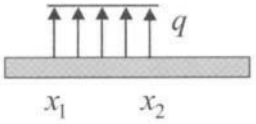
(L2) Torque



$f(x) = -\tau \dfrac{d}{dx}\delta(x - x_\tau)$

Load_Spec = [−1 x_τ τ]

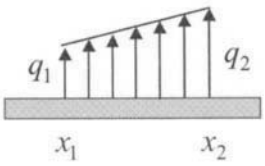
(D1) Uniform



$f(x) = q \quad \text{for } x_1 \le x \le x_2$

Load_Spec = [1 x_1 x_2 q]

(D2) Linear



$f(x) = q_1 + \dfrac{q_2 - q_1}{x_2 - x_1}(x - x_1) \quad \text{for } x_1 \le x \le x_2$

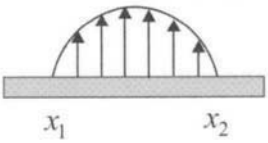
Load_Spec = [2 x_1 x_2 q_1 q_2]

(D3) Quadratic

$f(x) = c_0 + c_1 x + c_2 x^2 \quad \text{for } x_1 \le x \le x_2$

Load_Spec = [3 x_1 x_2 c_0 c_1 c_2]

(D4) Sinusoidal



$f(x) = A\sin(\omega x + \phi) \quad \text{for } x_1 \le x \le x_2$

Load_Spec = [4 x_1 x_2 A ω ϕ]

(ϕ in radians)

With `y_total` and `x`, the spatial distribution of the total response can be plotted by `plot(x, y_total)`.

Besides `beamcolumn`, function `beamcolumn1` can be called to plot the response of a beam-column at a particular point (see Window 2.7.)

Window 2.7. Function `beamcolumn1`

MATLAB Function: **`beamcolumn1`**

Purpose:
To compute the response of a beam-column at a particular point.

Synopsis:

```
beamcolumn1(xr, Load_Spec, p)
[y, p] = beamcolumn1(xr, Load_Spec, p)
```

Description:

`beamcolumn1(xr, Load_Spec, p)` plots the response (deflection, rotation, bending moment, and shear force) of the beam-column at point `xr` for a given axial load `p`. Here `Load_Spec` is a vector specifying a transverse load by Table 4.2.4. The value of `p` must be less than the Euler buckling load P_{cr} of the beam-column. By default, `xr` = $L/2$ where L is the column length and `p` = $0.5P_{cr}$.

`y = beamcolumn1(xr, Load_Spec, p)` returns the computed beam-column response at `xr` in a four-vector `y`, with `y(1)` being the deflection, `y(2)` the rotation, `y(3)` the bending moment, and `y(4)` the shear force.

EXAMPLE 2.4

In Fig. 4.2.12, a clamped-pinned beam-column of parameters EI = 30 and L = 4 is subject to a uniform transverse load q = 2.5 on its right half span. Let the axial load be P = 18. The response of the beam-column is plotted in Fig. 4.2.13 by the following commands:

```
>> setcolumn(30, 4, [2 1])
>> load = [1 2 4 2.5];
>> beamcolumn(load,18)
```

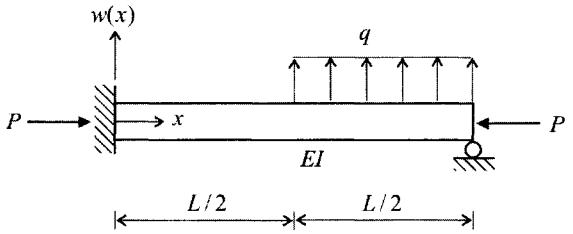


FIGURE 4.2.12

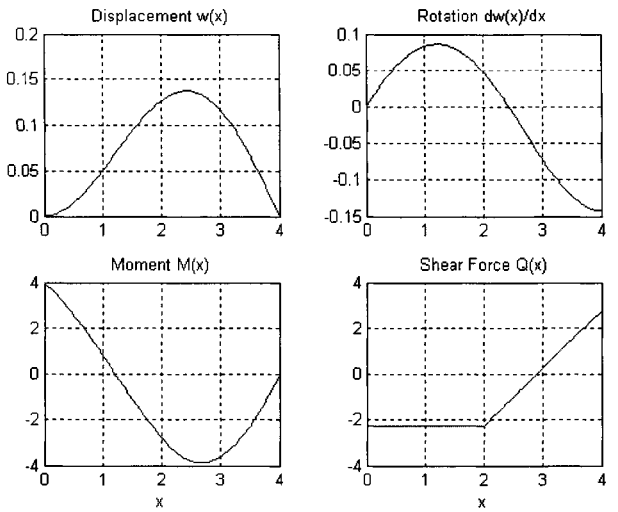


FIGURE 4.2.13

The Euler buckling load, by function `setcolumn`, is

$$P_{cr} = 37.8576$$

Now, with the same transverse load, examine the change of the beam-column response at different values of the axial load, say $P = 0$, 25, 30, and 35. By

```
>> [y1, x]=beamcolumn(load,0);
>> y2=beamcolumn(load, 25);
>> y3=beamcolumn(load, 30);
>> y4=beamcolumn(load, 35);
>> clf
>> plot(x, y1(1,:), x, y2(1,:), x, y3(1,:), x, y4(1,:))
>> grid; xlabel('x')
>> title('W(x) at P = 0, 25, 30 & 35')
```

the deflection curves of the beam-column are plotted in Fig. 4.2.14. It is seen that the deflection $w(x)$ dramatically increases with increase in the axial load.

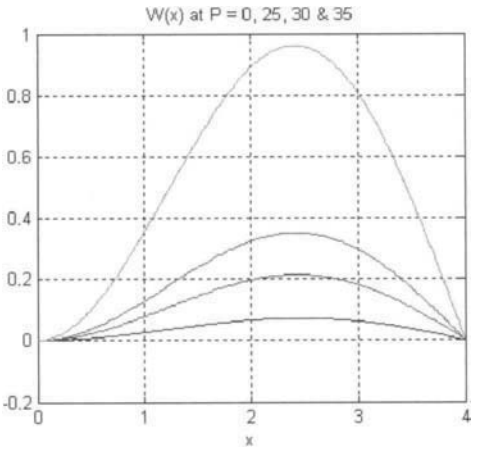


FIGURE 4.2.14

4.2.5 Geometric Imperfection

In design of structures, the effect of initial geometric imperfection of columns is a major concern. Consider a uniform column with an initial deflection curve $w_o(x)$ (see Fig. 4.2.15). Here, $w_o(x)$ appears without application of the axial load, and for this reason, the column is often referred to as geometrically imperfect or initially bent. Let $w(x)$ be the transverse deflection induced by the axial load P, measured from the initial bent geometry. The induced deflection is governed by the differential equation (Reference 3)

$$EI\frac{d^4w(x)}{dx^4} + P\frac{d^2w(x)}{dx^2} = -P\frac{d^2w_o(x)}{dx^2}. \tag{4.15}$$

The bending moment and shear force of the column are the same as given in Eq. (4.2). By comparing Eq. (4.15) with Eq. (4.14), one realizes that a geometrically imperfect column

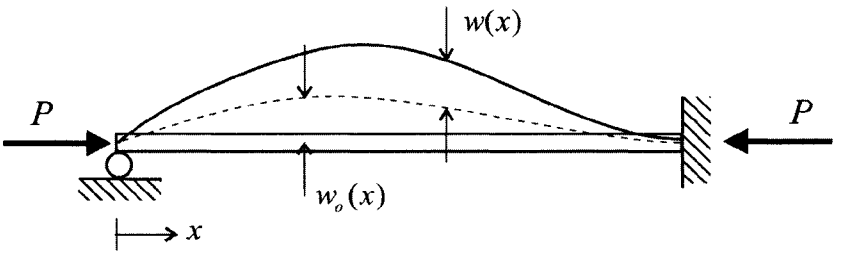


FIGURE 4.2.15 A column with initial geometric imperfection.

can be viewed as a beam-column that is subject to the "effective transverse load"

$$f(x) = -P\frac{d^2w_o(x)}{dx^2}. \tag{4.16}$$

Like a beam-column, a geometrically imperfect column does not suddenly buckle; it deflects immediately after a small axial load is applied. The deflection grows as the load increases and goes to infinity when the load approaches the Euler buckling load.

According to Eq. (4.16), functions `beamcolumn` and `beamcolumn1` given in Windows 2.6 and 2.7 can be directly applied to determine the deflection of an imperfect column; see the following examples.

EXAMPLE 2.5

A simply supported (pinned-pinned) column of parameters $EI = 1$ and $L = 1$ has the initial deflection curve $w_o(x) = \varepsilon \sin \pi x$. The effective load, by Eq. (4.16), is

$$f(x) = -P\frac{d^2w_o(x)}{dx^2} = P\varepsilon\pi^2 \sin \pi x.$$

The Euler buckling load of the column, by Eq. (4.9) and Table 4.2.2, is $P_{cr} = 9.8696$. Let the axial load be $P = 5$. Let the amplitude of the initial deflection be $\varepsilon = 0.01$. The deflection $w(x)$ of the column is plotted by

```
>> setcolumn(1, 1, [1 1])
>> e = 0.01; p = 5;
>> a = e*p*pi^2;
>> load = [4 0 1 a pi 0];
>> beamcolumn(load, p)
```

which yields Fig. 4.2.16.

Function `beamcolumn1` can be used to plot the deflection of the column at any point, versus the axial load. Consider different values of the initial deflection amplitude, say $\varepsilon = 0.01$, 0.02, and 0.05. The load-deflection curves at these ε values are generated by the following commands:

```
setcolumn(1, 1, [1 1])
npts = 99; xr = 0.5; pf = 9.8;
```

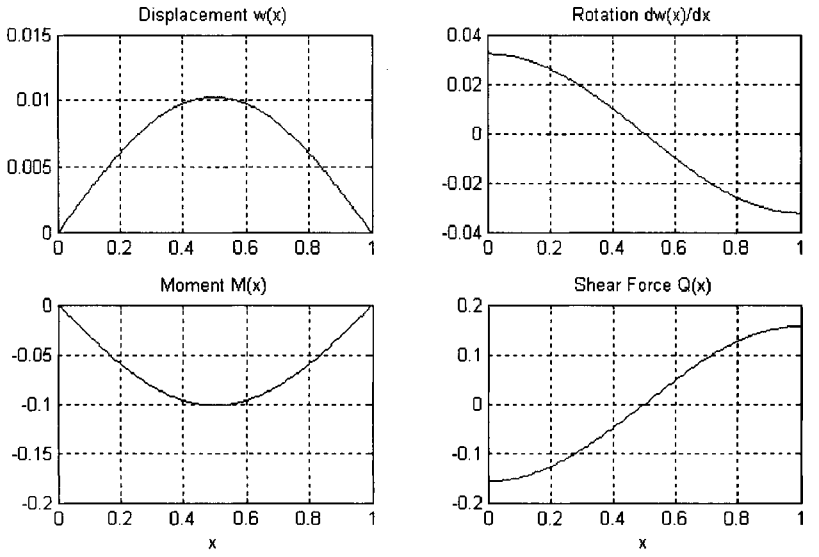


FIGURE 4.2.16

```
axial_load = linspace(0, 9.8, npts);
w = zeros(3, npts);
e = [0.01 0.02 0.05];
for j = 1:3
 e0 = e(j);
 for i = 1:npts
     p = axial_load(i)
     load = [4 0 1 e0*p*pi*pi pi 0];
     y = beamcolumn1(xr, load, p);
     w(j,i)  = y(1);
 end
end
Pcr = eigcolumn(1);
axial_load = axial_load/Pcr;
clf
plot(w(1,:), axial_load, w(2,:), axial_load, w(3,:), axial_load)
grid; title('Load-deflection curves')
xlabel('W at x = L/2')
ylabel('P/Pcr')
```

which produce the plot in Fig. 4.2.17.

One convenient way to execute the above commands is to put them in a script M-file and then type the name of the M-file from the MATLAB command window. (See Appendix B.4 for how to create such an M-file.) As can be seen, with the geometric imperfection $w_o(x)$, the deflection of the column appears immediately after the axial load is applied, and it increases rapidly as the axial load approaches the Euler buckling load P_{cr}.

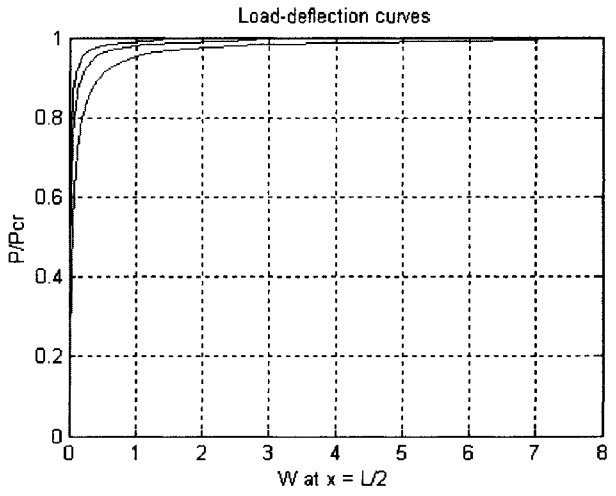


FIGURE 4.2.17

EXAMPLE 2.6

The clamped-pinned column in Fig. 4.2.18 has a localized initial deflection

$$w_o(x) = \begin{cases} \frac{16\varepsilon}{a^4}(x - x_1)^2 (x - x_2)^2 & \text{for } x_1 \le x \le x_2 \\ 0 & \text{otherwise} \end{cases}$$

where $x_1 = (L - a)/2$ and $x_2 = (L + a)/2$. The effective transverse load, by Eq. (4.16), is

$$f(x) = -P\frac{d^2 w_o(x)}{dx^2} = \begin{cases} c_2 x^2 + c_1 x + c_0 & \text{for } x_1 \le x \le x_2 \\ 0 & \text{otherwise} \end{cases}$$

with

$$c_2 = -192\frac{P\varepsilon}{a^4}, \quad c_1 = 192\frac{P\varepsilon}{a^4}(x_1 + x_2), \quad c_0 = -32\frac{P\varepsilon}{a^4}(x_1^2 + 4x_1 x_2 + x_2^2).$$

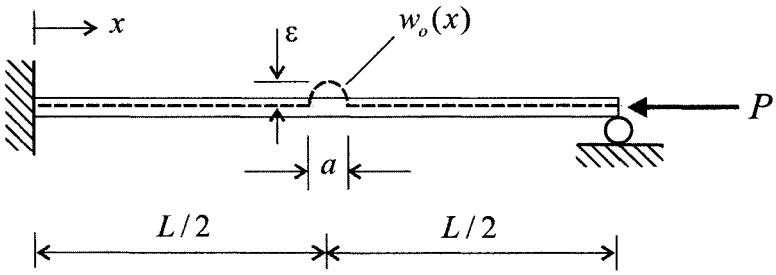


FIGURE 4.2.18

Assume that the column parameters are $EI = 25$ and $L = 4$. The Euler buckling load is $P_{cr} = 31.548$. Let the axial load be $P = 25$. The spatial distribution of the column

response is plotted in Fig. 4.2.19 by

```
>> setcolumn(25, 4, [2 1])
>> a = 0.2; x1 = (4-a)/2; x2 = (4+a)/2;
>> p = 25; e0 = 0.05;
>> c2 = -192*p*e0/a^4;
>> c1 = -c2*(x1+x2);
>> c0 = -32*p*e0/a^4*(x1^2 + x2^2 + 4*x1*x2);
>> load = [3 x1 x2 c2 c1 c0];
>> beamcolumn(load, p, 1001)
```

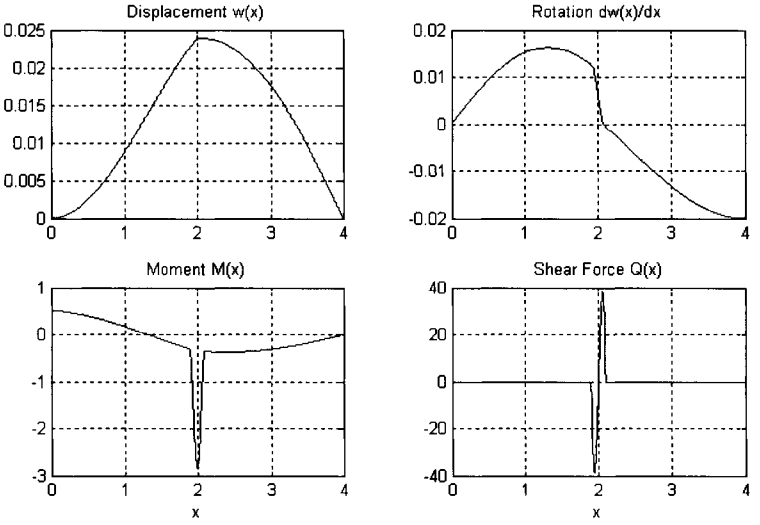


FIGURE 4.2.19

Also, following Example 2.5, the load-deflection curves of the imperfect column at $x = L/2$, for $\varepsilon = 0.02$, 0.05, and 0.1, are plotted in Fig. 4.2.20, by the following commands:

```
setcolumn(25, 4, [2 1])
Pcr = eigcolumn(1);
npts = 50; e = [0.02 0.05 0.1]; xr = 2;
a = 0.2; x1 = (4-a)/2; x2 = (4+a)/2;
axial_load = linspace(0, (npts-1)*Pcr/npts, npts);
w = zeros(3, npts);
for j = 1:3
 e0 = e(j);
 for i = 1:npts
     p = axial_load(i);
     c2 = -192*p*e0/a^4;
     c1 = -c2*(x1+x2);
     c0 = -32*p*e0/a^4*(x1^2 + x2^2 + 4*x1*x2);
     load = [3 x1 x2 c2 c1 c0];
     y = beamcolumn1(xr, load, p);
     w(j,i) = y(1);
```

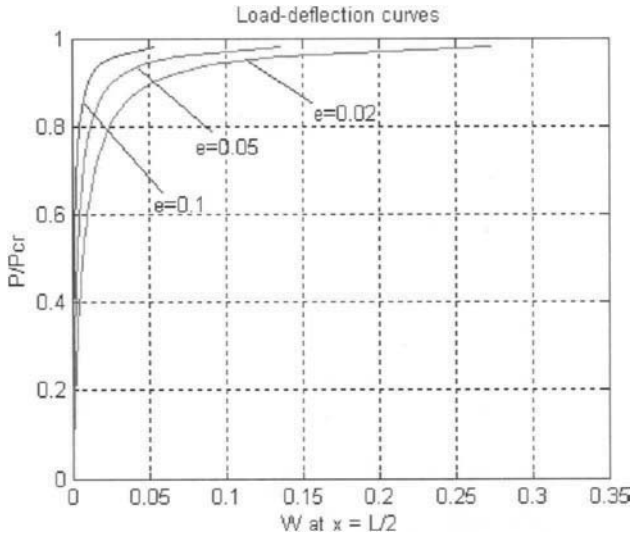


FIGURE 4.2.20

```
 end
end
axial_load = axial_load/Pcr;
clf
plot(w(1,:), axial_load, w(2,:), ...
        axial_load, w(3,:), axial_load)
grid; title('Load-deflection curves');
xlabel('W at x = L/2'); ylabel('P/Pcr')
```

4.3 Stepped and Nonuniform Columns

The buckling analysis in the previous section is about uniform columns. This section is concerned with buckling of two types of nonuniform columns: constrained stepped columns and single-body columns with bending stiffness being a function of the spatial coordinate x. Buckling solutions of these columns are obtained by the Distributed Transfer Function Method discussed in Section 4.4.

4.3.1 Constrained Stepped Columns

The constrained stepped column in consideration is an axially loaded structure of n uniform Euler-Bernoulli beam segments that are serially connected at n-1 points called *nodes* (see Fig. 4.3.1). Those nodes may be constrained by translational and torsional springs (see node 3), and hinge and sliding constraints (see node n-1). Also, some beam segments may be constrained by elastic foundation or distributed spring (k_e) within their domains (see the segment between nodes 1 and 2). The two ends (boundaries) of the column are called nodes 0 and n.

Denote the jth beam segment by S_j, which is bounded by nodes j-1 and j. The transverse deflection $w_j(x)$ of S_j is governed by

$$\left(EI_j\frac{d^4}{dx^4} + P\frac{d^2}{dx^2} + k_j\right)w_j(x) = 0, \quad 0 \le x \le L_j \tag{4.17}$$

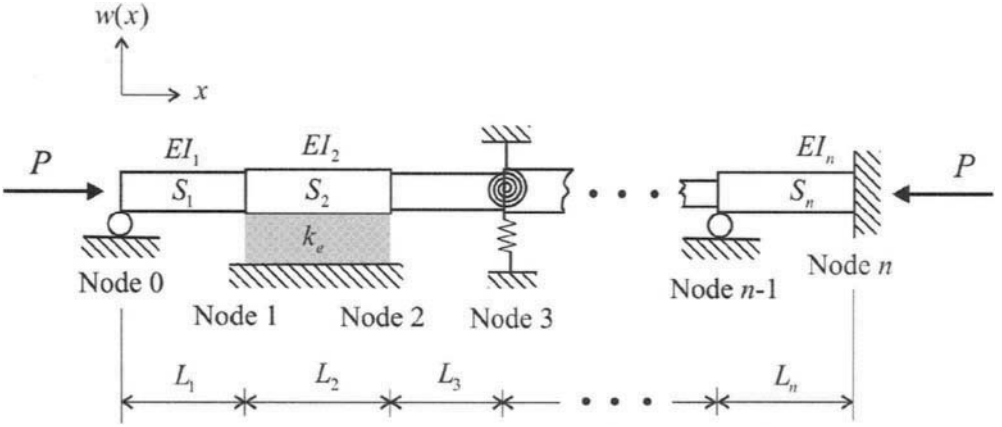


FIGURE 4.3.1 A constrained stepped column.

where EI_j and L_j are the bending stiffness and length of the segment, k_j is the coefficient of elastic foundation, and x is the local spatial coordinate of S_j measured from its left end. The boundary conditions of the column are specified at its ends (nodes 0 and n), and they are the same as given in Table 4.2.1.

The matching or compatibility conditions at node j, where two adjacent segments S_j and S_{j+1} are interconnected, are written as

$$
\begin{aligned}
w_{j+1}(0) &= w_j(L_j) \\
\frac{d}{dx}w_{j+1}(0) &= \frac{d}{dx}w_j(L_j) \\
EI_{j+1}\frac{d^2}{dx^2}w_{j+1}(0) &= EI_j\frac{d^2}{dx^2}w_j(L_j) + \tau_{c,j} \\
EI_{j+1}\frac{d^3}{dx^3}w_{j+1}(0) &= EI_j\frac{d^3}{dx^3}w_j(L_j) - p_{c,j}
\end{aligned}
\tag{4.18}
$$

where $\tau_{c,j}$ and $p_{c,j}$ are the constraint torque and force at the node. If the column at the node is constrained by a rotational spring of coefficient k_r and a translational spring of coefficient k_t,

$$
\tau_{c,j} = k_r\frac{d}{dx}w_j(L_j), \quad p_{c,j} = k_t w_j(L_j).
$$

If a hinge is at the node, $w_j(L_j) = 0$, $\tau_{c,j} = 0$, and the constraint force $p_{c,j}$ is an unknown to be determined. If a sliding constraint is at the node, $\frac{d}{dx}w_j(L_j) = 0$, $p_{c,j} = 0$, and the constraint torque $\tau_{c,j}$ is an unknown.

Equations (4.17) and (4.18) along with boundary conditions form an eigenvalue problem of the column, with the buckling load as an eigenvalue. The Toolbox of this chapter provides functions `setcolumn2` and `plotmsh2` for computing and displaying the eigensolutions; see Windows 3.1 and 3.2.

Window 3.1. Function setcolumn2

MATLAB Function: **setcolumn2**

Purpose:
To compute the buckling loads and mode shapes of a constrained stepped column.

Synopsis:

```
setcolumn2(Beam_Spec, BC_Spec, Constr_Spec)
setcolumn2(Beam_Spec, BC_Spec, Constr_Spec, n_mode)
```

Description:
`setcolumn2(Beam_Spec, BC_Spec, Constr_Spec)` computes the first four buckling loads and mode shapes of the column, where `Beam_Spec` is a matrix consisting of the parameters of beam segments; `BC_Spec` is a vector specifying the boundary conditions of the column; and `Constr_Spec` is a matrix specifying spring constraints at any of nodes 1 to n-1. Assignment of these arguments is given on the next page. If the column has no constrains, simply use `setcolumn2(Beam_Spec, BC_Spec)`.

`setcolumn2(Beam_Spec, BC_Spec, Constr_Spec, n_mode)` specifies that the first `n_mode` buckling modes are computed. By default, `n_mode` = 4. If the column has no constrains, set `Constr_Spec = 0`.

Window 3.2. Function plotmsh2

MATLAB Function: **plotmsh2**

Purpose:
To plot a buckling mode shape of a stepped or nonuniform column.

Synopsis:

```
plotmsh2(k)
plotmsh2(k, opt, npts)
[y, x] = plotmsh2(k, opt, npts)
```

Description:
`plotmsh2(k)` plots the spatial distribution of the kth buckling mode shape (deflection).

`plotmsh(k, opts, npts)` plots the kth buckling mode shape with the following options:

`opt` = 1	plot deflection $w(x)$ only
`opt` = 2	plot slope $dw(x)/dx$ only
`opt` = 3	plot bending moment $M(x)$ only
`opt` = 4	plot shear force $Q(x)$ only
`opt` = 0	plot all above four curves (deflection, slope, moment, and shear force)

Window 3.2. Function `plotmsh2` (*Continued*)

Here `npts` is the number of equally-spaced points along each beam segment. By default, `opt` = 1, and `npts` = 101 for a stepped column and `npts` = 11 for a nonuniform column.

`[y, x] = plotmsh(k, opts, npts)` returns the response of the column in the kth mode in a four-row matrix `y` and spatial points of computation in a vector `x`. Here, `y(1,:)` stores the deflection $w(x)$, `y(2,:)` the rotation $dw(x)/dx$, `y(3,:)` the bending moment $M(x)$, and `y(4,:)` the shear force $Q(x)$. With `y` and `x`, the spatial distribution of the modal response can be plotted by the MATLAB function `plot`. For example, for the deflection $w(x)$, use `plot(x, y(1,:))`.

In Window 3.1, **`setcolumn2`** assigns the parameters, boundary conditions, and constraints for an n-span column by the following arguments:

(i) The parameters of the beam segments are assigned by an $n \times 2$ matrix

$$\texttt{Beam\_Spec} = \begin{bmatrix} EI_1 & L_1 \\ EI_2 & L_2 \\ \vdots & \vdots \\ EI_n & L_n \end{bmatrix} \tag{4.19}$$

where the ith row contains the bending stiffness and length of the ith segment. If some of the segments are constrained by elastic foundations within their domains, `Beam_Spec` is an $n \times 3$ matrix

$$\texttt{Beam\_Spec} = \begin{bmatrix} EI_1 & L_1 & k_1 \\ EI_2 & L_2 & k_2 \\ \vdots & \vdots & \vdots \\ EI_n & L_n & k_n \end{bmatrix} \tag{4.20}$$

where k_j is the coefficient of elastic foundation on the jth segment. If no distributed spring is imposed on the jth segment, k_j is set to zero.

(ii) The boundary conditions of the column (at nodes 0 and n) are assigned by a row vector `BC_Spec` according to Table 4.2.3.

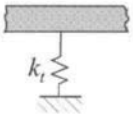
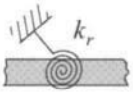
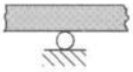
(iii) The constraints at nodes 1 to n-1 are assigned by an `Nc` $\times$ 3 matrix

$$\texttt{Constr\_Spec} = \begin{bmatrix} \texttt{Constr\_1} \\ \texttt{Constr\_2} \\ \vdots \\ \texttt{Constr\_Nc} \end{bmatrix} \tag{4.21}$$

where `Nc` is the total number of constraints. The jth row of the matrix specifies the jth spring constraint, and is of the form

```
Constr_j  = [node_no constr_type_no ks]
```  (4.22)

TABLE 4.3.1 Specification of Constraints

Constraint Type		Constr_j
(C1) Translational spring		[node_no 1 k_t]
(C2) Torsional spring		[node_no 2 k_r]
(C3) Hinge		[node_no 3]
(C4) Sliding constraint		[node_no 4]

where node_no is node number, constr_type_no is an integer from 1 to 4, and ks is a spring coefficient for a spring constraint or zero for a hinge or sliding constraint. See Table 4.3.1 for the entries of Constr_j.

Note. For the information of a stepped column structure that is set up by function setcolumn2, type **systinfo2** in the MATLAB command window.

EXAMPLE 3.1

In Fig. 4.3.2, a two-stepped column is constrained by a translational spring at the midpoint. Let the parameters of the structure be $EI_1 = 5, EI_2 = 10, L = 1$. Let the value of the spring coefficient be $k_t = 5$. The commands:

```
>> beam = [5 1; 10 1];
>> bc = [1 2];
>> constr = [1 1 5];
>> setcolumn2(beam, bc, constr)
```

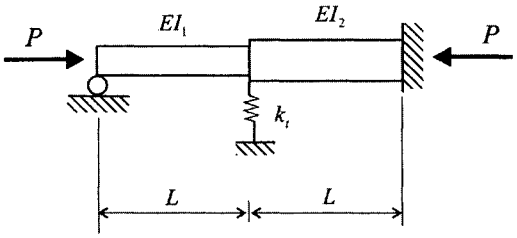


FIGURE 4.3.2

yield the first four buckling loads of the column

```
Mode 1:  P = 32.8277
Mode 2:  P = 104.1935
Mode 3:  P = 204.6129
Mode 4:  P = 333.215
```

and the mode shapes plotted in Fig. 4.3.3.

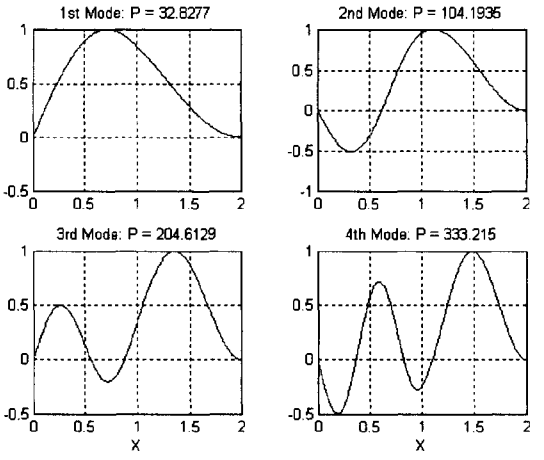


FIGURE 4.3.3

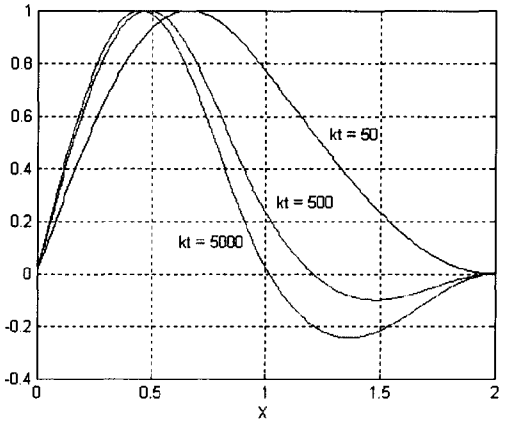


FIGURE 4.3.4

Now set the value of the spring coefficient to $k_t = 50$, 500, and 5000. The first buckling mode shape of the column at those k_t values is plotted by

```
>> setcolumn2(beam, bc, [1 1 50], 1)
>> [y1, x] = plotmsh2(1);
>> setcolumn2(beam, bc, [1 1 500], 1)
>> y2 = plotmsh2(1);
```

```
>> setcolumn2(beam, bc, [1 1 5000], 1)
>> y3 = plotmsh2(1);
>> plot(x, y1(1,:), x, y2(1,:), x, y3(1,:))
>> grid; xlabel('X')
```

as shown in Fig. 4.3.4. It is seen that the deflection of the column at the midpoint approaches zero as the spring coefficient goes to infinity. In that case, the spring becomes a hinge.

EXAMPLE 3.2

In Fig. 4.3.5, a clamped-free uniform column is constrained by a hinge at the midpoint and an elastic foundation along its right-half span. The column parameters are $EI = 1$, $L = 2$, and $k_s = 4.5$. In the buckling analysis, the column is divided into two segments of equal length. The commands

```
>> beam = [1 1 0; 1 1 4.5];
>> bc = [2 3];
>> constr = [1 3];
>> setcolumn2(beam, bc, constr)
```

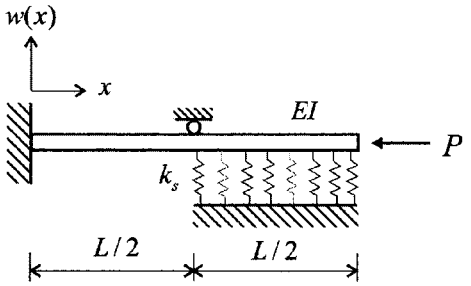


FIGURE 4.3.5 A column on an elastic foundation.

yield the first four buckling loads

```
Mode 1: P = 2.62
Mode 2: P = 13.6211
Mode 3: P = 30.3339
Mode 4: P = 49.9488
```

and the associate mode shapes plotted in Fig. 4.3.6.

Furthermore, to plot modal moment M(x) and shear force Q(x) of the first buckling mode, type in the MATLAB command window

```
>> plotmsh2(1, 0)
```

which produces Fig. 4.3.7. Note the jump (discontinuity) in the shear force distribution at the midpoint of the column, which is due to the constraint force by the hinge.

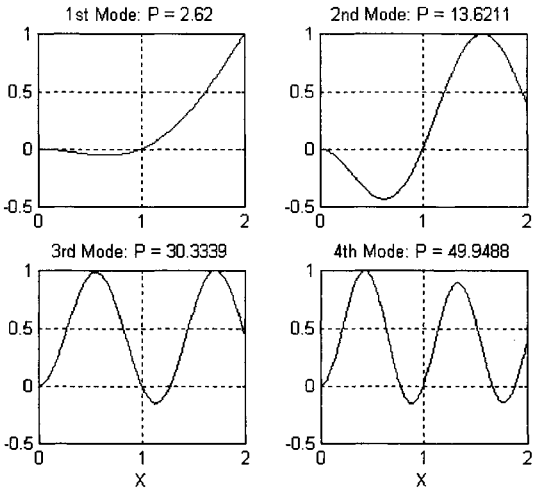


FIGURE 4.3.6

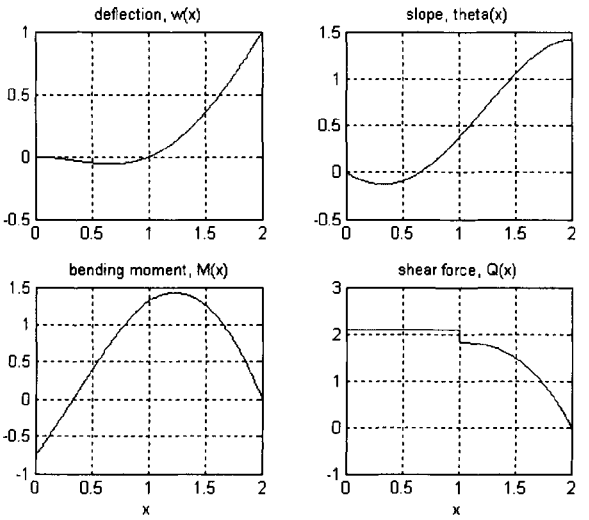


FIGURE 4.3.7

Function `setcolumn2` can also be used to investigate a column of very small bending stiffness on an elastic foundation. The buckling of such a column can exhibit mode shapes of high wave numbers; see Example 3.3.

4.3.2 Nonuniform Columns

Consider a nonuniform column with bending stiffness $EI(x)$ as a function of the spatial coordinate x. The buckling deflection $w(x)$ of the column is governed by

$$\frac{d^2}{dx^2}\left(EI(x)\frac{d^2}{dx^2}w(x)\right) + P\frac{d^2}{dx^2}w(x) = 0, \quad 0 \le x \le L. \tag{4.23}$$

EXAMPLE 3.3

Consider a clamped-guided uniform column on an elastic foundation; see Fig. 4.3.8. The buckling deflection of the column is governed by the differential equation

$$\left(EI\frac{d^4}{dx^4} + P\frac{d^2}{dx^2} + k_e\right) w(x) = 0, \quad 0 \le x \le L$$

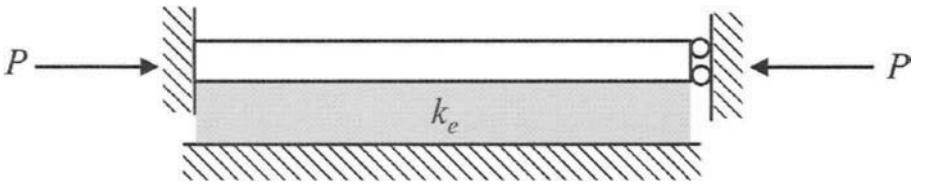


FIGURE 4.3.8

where k_e is the coefficient of the elastic foundation. Let the column parameters be $EI = 10^{-5}$, $L = 1$, and $k_e = 1$. Note that the bending stiffness of the column is very small, compared to $k_e L^4$. Thus, the elastic restoring force of the column in the transverse direction mainly comes from the elastic foundation. By

```
>> EI = 0.00001; L = 1; ks = 1;
>> setcolumn2([EI L ks], [2 4])
```

the buckling mode shapes of the column are plotted in Fig. 4.3.9, as compared to the buckling modes of the column without the elastic foundation given in Fig. 4.3.10. Here, the elastic foundation raises the buckling loads, but at the same time increases the wave numbers of the mode shapes. This high wave number of buckling modes somewhat resembles the wrinkling pattern of a column of weak bending stiffness, although geometric nonlinearities are involved there.

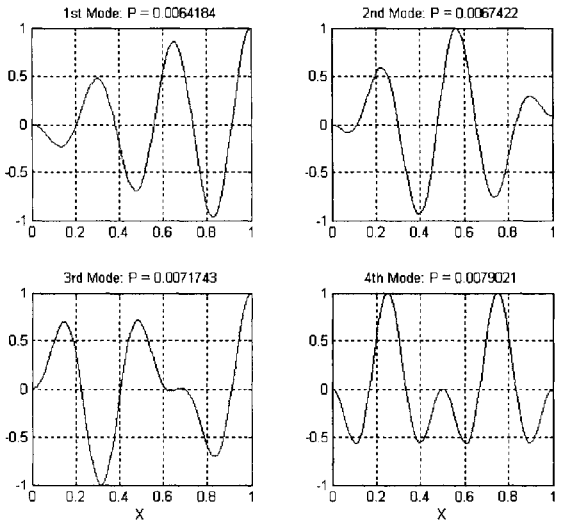


FIGURE 4.3.9 Buckling mode shapes of the column with elastic foundation.

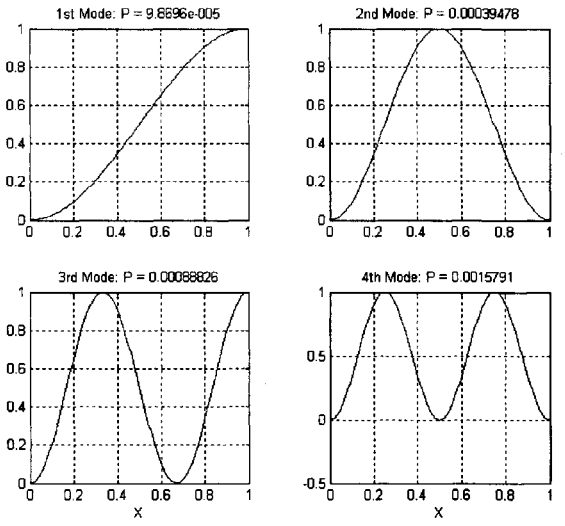


FIGURE 4.3.10 Buckling mode shapes of the column without elastic foundation.

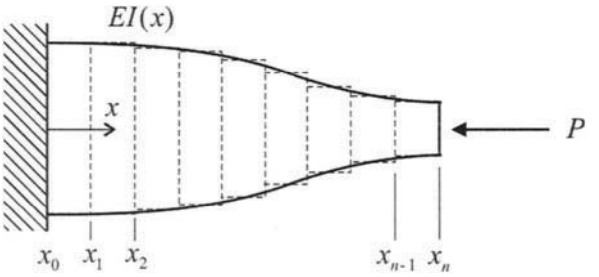


FIGURE 4.3.11 Approximation of a nonuniform column by a stepped column.

Exact buckling solutions for nonuniform columns are limited to a few special cases. In general, approximate solution methods have to be used.

In this section, the buckling problem described by Eq. (4.23) is solved in two steps: (i) the nonuniform column is approximated as a stepped column; and (ii) the buckling solutions of the stepped column are determined following Section 4.3.1. Figure 4.3.11 shows an approximation of the nonuniform column by an n-step column. Let the nodes of the stepped column in the global coordinate be at $x_j, j = 0, 1, 2, \ldots, n$. The transverse deflection of the jth segment is governed by Eq. (4.17). The bending stiffness of the jth segment is assigned as

$$EI_j = \frac{1}{2}\left(EI(x_{j-1}) + EI(x_j)\right) \tag{4.24}$$

where $EI(x_j)$ is the value of the nonuniform bending stiffness at node j. With the above approximation, the buckling problem is readily solved following Section 4.3.1. As the number n of segments increases, the solutions of the stepped column converge to those of the original nonuniform column.

The Toolbox of this chapter provides a function `nonucolumn` for determining buckling solutions of nonuniform columns by the above described method; see Window 3.3.

Window 3.3. Function nonucolumn

MATLAB Function: **nonucolumn**

Purpose:
To compute the buckling loads and mode shapes of a nonuniform column.

Synopsis:

```
nonucolumn(EI_Spec, L, BC_Spec)
nonucolumn(EI_Spec, L, BC_Spec, n_mode, n_seg)
```

Description:
`nonucolumn(EI_Spec, L, BC_Spec)` computes the first four buckling loads and mode shapes of the column, where `EI_Spec` is a vector or matrix specifying the nonuniform bending stiffness by Table 4.3.2, `L` is the length of the column, and `BC_Spec` is a vector specifying the boundary conditions of the column by Table 4.2.3.

`nonucolumn(EI_Spec, L, BC_Spec, n_mode, n_seg)` computes the first `n_mode` buckling loads and mode shapes, where `n_seg` is the number of beam segments used in approximating the nonuniform column. By default, `n_mode` = 4, and `n_seg` = 20. Both `n_mode` and `n_seg` can be replaced by [ ] as a place holder for their default values.

Note 1. For the first six types of stiffness distribution in Table 4.3.2,

```
EI_Spec = [type_no par1 par2 ...]
```

where `type_no` is an integer between 1 and 6 identifying the type of stiffness distribution, and `par1 par2 ...` are parameters describing the distribution. For Type 7, `EI_Spec` is a two-row matrix. Types 6 and 7 (`type_no` = 6 or 7) allow the user to specify any nonuniform bending stiffness. Specification of the boundary conditions for a nonuniform column is the same as given in Table 4.2.3.

Note 2. For the information of a nonuniform column that is set up by function `nonucolumn`, type **`systinfo2`** in the MATLAB command window.

Note 3. When a bending stiffness distribution of Type 6 or 7 is used, the assignment of parameter `n_seg` as an input argument of function `nonucolumn` has no effect.

Note 4. Function `plotmsh2` in Window 3.2 can be used to plot buckling mode shapes for a nonuniform column, provided that function `nonucolumn` has been called. In this case, the number `npts` of spatial points selected on each segment for computation is 11 by default.

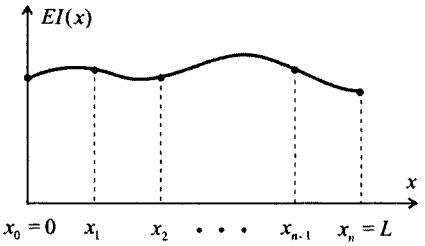
EXAMPLE 3.4

Consider a pinned-clamped column with bending stiffness

$$EI(x) = EI_0\,(1 - \alpha x/L)^3$$

where L is the length of the column. Let the column parameters be $EI_0 = 2, L = 4$, $\alpha = 0.3$.

TABLE 4.3.2 Specification of Bending Stiffness $EI(x)$ by Vector `EI_Spec`

Distribution of Bending Stiffness	**EI_Spec**
Type 1 (linear or tapered) $EI(x) = EI_0\left(1 - \frac{x}{L}\right) + EI_L\frac{x}{L}$	`EI_Spec = [1  EI0  EIL]`
Type 2 (polynomial) $EI(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n$	`EI_Spec = [2  c0  c1  c2  ...  cn]`
Type 3 (sinusoidal) $EI(x) = a + b\sin(cx + \theta_0)$	`EI_Spec = [3  a  b  c  θ0]` θ_0 given in degrees
Type 4 $EI(x) = a[c_0 + c_1x + \cdots + c_nx^n]^b$ b – an arbitrary number	`EI_Spec = [4  a  b  c0  c1  ...  cn]`
Type 5 (exponential) $EI(x) = a + be^{cx}$	`EI_Spec = [5  a  b  c]`
Type 6 (user-provided) 	`EI_Spec = [6  EI(x0)  EI(x1)  ...  EI(xn)]` Here $EI(x_0)$, $EI(x_1)$,..., $EI(x_n)$ are the values of $EI(x)$ at the n+1 *equally-spaced* points $x_k = k\Delta$, $k = 0, 1, \ldots, n$, $\Delta = L/n$
Type 7 (user-provided) The distribution of $EI(x)$ is the same as shown in Type 6, except that the points $x_0, x_1, \ldots, x_n$ don't have to be equally spaced.	$\texttt{EI\_Spec} = \begin{bmatrix} EI(x_0) & EI(x_1) & \cdots & EI(x_n) \\ x_0 & x_1 & \cdots & x_n \end{bmatrix}$ Here $EI(x_0)$, $EI(x_1)$,..., $EI(x_n)$ are the values of $EI(x)$ at the n+1 points x_k, $k = 0, 1, \ldots, n$, with $x_0 = 0$ and $x_n = L$.

```
>> EI0 = 2; e = 0.3; L = 4;
>> EI_Spec = [4 EI0 3 1 -e/L];
>> nonucolumn(EI_Spec, L, [1 2])
```

yields the buckling loads of the column

```
Mode 1: P = 1.4918
Mode 2: P = 4.408
Mode 3: P = 8.7815
Mode 4: P = 14.6127
```

and the mode shapes plotted in Fig. 4.3.12.

TABLE 4.3.3

Number of Segments	*1st Buckling Load*	*2nd Buckling Load*	*3rd Buckling Load*
5	1.5102	4.4697	8.9358
10	1.4953	4.4188	8.8041
20	1.4918	4.408	8.7815
50	1.4908	4.4051	8.7756
100	1.4906	4.4047	8.7748

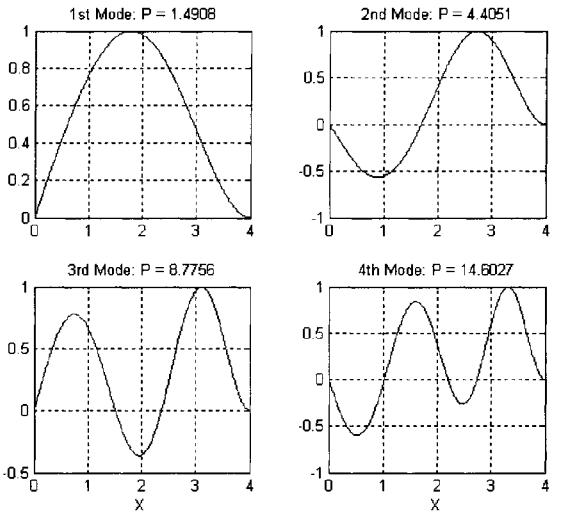


FIGURE 4.3.12

To check the accuracy and convergence of the solution method, the first three buckling loads are recalculated with the number of segments `n_seg` set to 5, 10, 20, 50, and 100. The computed results are listed in Table 4.3.3. As can be seen, the results obtained by 10 segments are accurate enough, with less than 0.5% of deviation from those obtained by 100 segments.

EXAMPLE 3.5

In Fig. 4.3.13, a clamped-pinned column is composed of three elements. The left and right elements are uniform with constant bending stiffness EI_1 and EI_3, respectively. The middle element is tapered with bending stiffness as a linear function of x

$$EI_2(x) = EI_1\left(1 - \frac{x-a}{b}\right) + EI_3\frac{x-a}{b} \quad \text{for } a \le x \le a+b.$$

Let the column parameters be $EI_1 = 50$, $EI_3 = 30$, $a = 4$, $b = 3$, $c = 2$. The column can be analyzed by function `nonucolumn` using Type 6 stiffness distribution given in Table 4.3.2.

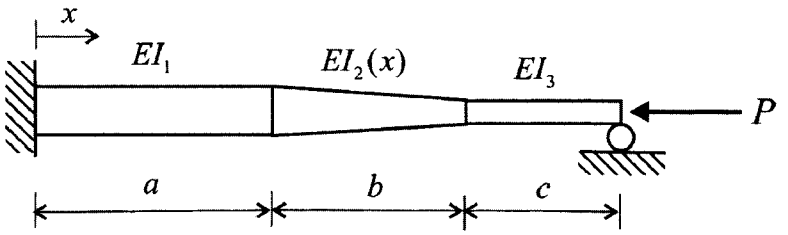


FIGURE 4.3.13

```
>> a = 4; b = 3; c= 2;
>> L = a+b+c;
>> EI_a= 50; EI_b = 30;
>> stiffness_1 = EI_a*ones(1,41);
>> xx = [41:69]/10;
>> stiffness_2 = EI_a*(1-(xx-a)/b) + EI_b*(xx-a)/b;
>> stiffness_3 = EI_b*ones(1,21);
>> EI_Spec = [6 stiffness_1 stiffness_2 stiffness_3];
>> nonucolumn(EI_Spec, L, [2 1])
```

produces the following buckling loads

```
Buckling Loads
--------------------
Mode 1: P = 9.8542
Mode 2: P = 29.9955
Mode 3: P = 59.8228
Mode 4: P = 99.7793
```

In addition,

```
>> plotmsh2(1)
```

plots the first buckling mode shape in Fig. 4.3.14.

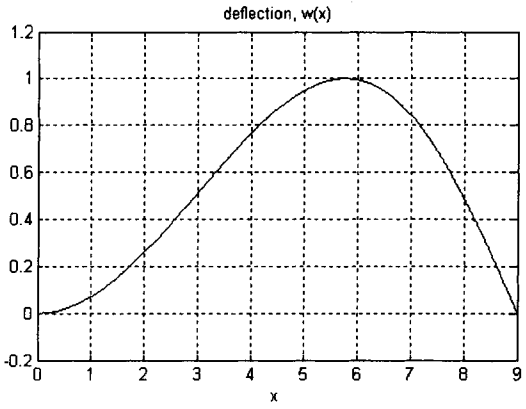


FIGURE 4.3.14

4.4 The Distributed Transfer Function Method

This section presents the Distributed Transfer Function Method (DTFM), which is used to develop the MATLAB functions presented in the previous sections. For more information on the DTFM, refer to Appendix C and Reference 4. The reader who is only interested in using the Toolbox can skip this section.

Spatial Space Formulation

In a DTFM-based buckling analysis, Eq. (4.1) is cast in the equivalent spatial state form

$$\frac{d}{dx}\{\eta(x)\} = [F]\{\eta(x)\} \tag{4.25}$$

where

$$\{\eta(x)\} = \begin{pmatrix} w(x) \\ w'(x) \\ w''(x) \\ w'''(x) \end{pmatrix}, \quad [F] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\beta^2 & 0 \end{bmatrix} \tag{4.26}$$

with $w' = dw/dx$ and $\beta = \sqrt{P/EI}$. For the beam-column equation (4.14), its equivalent state form is

$$\frac{d}{dx}\{\eta(x)\} = [F]\{\eta(x)\} + \frac{1}{EI}f(x)\{e_4\} \tag{4.27}$$

where $\{e_4\} = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T$. The boundary conditions of a column or beam-column that are listed in Table 4.2.1 can also be written in a state form

$$[M_b]\{\eta(0)\} + [N_b]\{\eta(L)\} = 0 \tag{4.28}$$

where $[M_b]$ and $[N_b]$ are four-by-four matrices. For instance, for a clamped-free column,

$$[M_b] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad [N_b] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & P & 0 & EI \end{bmatrix}$$

With the above formulation, the problems of columns and beam-columns that are discussed in Sections 4.2 and 4.3 can be conveniently solved.

Eigensolutions of Uniform Columns

The buckling problem of a column described in Section 4.2.1 is redefined by Eq. (4.25) with the boundary condition (4.28). Write the solution of Eq. (4.25) as

$$\eta(x) = e^{[F]x}\eta(0) \tag{4.29}$$

where $e^{[F]x}$ is the exponential of the matrix $[F]x$. It can be shown that

$$e^{[F]x} = \begin{bmatrix} 1 & x & \frac{1}{\beta^2}(1-\cos\beta x) & \frac{1}{\beta^3}(\beta x - \sin\beta x) \\ 0 & 1 & \frac{1}{\beta}\sin\beta x & \frac{1}{\beta^2}(1-\cos\beta x) \\ 0 & 0 & \cos\beta x & \frac{1}{\beta}\sin\beta x \\ 0 & 0 & -\beta\sin\beta x & \cos\beta x \end{bmatrix} \tag{4.30}$$

Substituting Eq. (4.29) into Eq. (4.28) yields the homogeneous equation

$$\left([M_b] + [N_b]\, e^{[F]L}\right)\{\eta(0)\} = 0. \tag{4.31}$$

The characteristic equation of the column then is

$$\Delta(P) \equiv \det\left([M_b] + [N_b]\, e^{[F]L}\right) = 0 \tag{4.32}$$

where $\Delta(P)$ is a transcendental function with an infinite number of zeros. The characteristic equation can be accurately and easily solved by standard root-searching techniques. The roots $P_1, P_2, \ldots$ are the buckling loads (eigenvalues) of the column, with the smallest P_1 being the Euler buckling load P_{cr}. The buckling mode shape (eigenfunction) associated with a buckling load P_k is obtained in two steps. First, substitute P_k into Eq. (4.31) to obtain a nonzero solution of the vector $\{\eta(0)\}$. Second, with the computed $\{\eta(0)\}$, obtain the modal displacement and internal forces by

$$\begin{pmatrix} w(x) \\ M(x) \\ Q(x) \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & EI & 0 \\ 0 & P & 0 & EI \end{bmatrix} e^{[F]x}\eta(0) \tag{4.33}$$

Thus, with the space formulation, exact analytical solutions of the eigenvalue problem are obtained.

Response of Beam-Columns

The beam-column problem discussed in Section 4.2.4 can be described by the state equation (4.27) with the boundary conditions (4.28). The solution of Eq. (4.27), by the DTFM, is of the form

$$\{\eta(x)\} = \frac{1}{EI}\int_0^L G(x,\xi) f(\xi) d\xi \ \{e_4\} \tag{4.34}$$

where $G(x,\xi)$ is the Green's function of the beam-column, and is given by

$$G(x,\xi) = \begin{cases} e^{[F]x}\left([M_b] + [N_b]\, e^{[F]L}\right)^{-1}[M_b]\, e^{-[F]\xi} & \text{for } x \geq \xi \\ -e^{[F]x}\left([M_b] + [N_b]\, e^{[F]L}\right)^{-1}[N_b]\, e^{[F](L-\xi)} & \text{for } x \leq \xi \end{cases} \tag{4.35}$$

Because $G(x,\xi)$ is in a closed form, the integral in Eq. (4.34) can be determined by exact quadrature. Thus, the response of the beam-column with arbitrary boundary conditions is obtained in exact and closed form

$$\begin{pmatrix} w(x) \\ M(x) \\ Q(x) \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & EI & 0 \\ 0 & P & 0 & EI \end{bmatrix} \eta(x) \tag{4.36}$$

Response of Eccentrically Loaded Columns

The governing equation of a column subject to an axial load P with eccentricity e is the same as Eq. (4.25). The boundary conditions of the column, however, involve an inhomogeneous term due to the end moment Pe:

$$[M_b]\{\eta(0)\} + [N_b]\{\eta(L)\} = Pe\{\gamma_b\} \tag{4.37}$$

where $\{\gamma_b\}$ is a constant vector. For instance, for the column shown in Fig. 4.2.5, the boundary conditions are given by Eq. (4.37) with

$$[M_b] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & EI & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad [N_b] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & EI & 0 \\ k_t & -P & 0 & -EI \end{bmatrix}, \quad \{\gamma_b\} = \begin{pmatrix} 0 \\ -1 \\ -1 \\ 0 \end{pmatrix}$$

The solution of Eq. (4.25) subject to the boundary conditions (4.37), by the DTFM, is given by

$$\eta(x) = H(x)\{\gamma_b\}Pe \tag{4.38}$$

where $H(x)$ is the boundary influence function of the column, and is of the form

$$H(x) = e^{[F]x}\left([M_b] + [N_b]e^{[F]L}\right)^{-1}. \tag{4.39}$$

With Eqs. (4.38) and (4.36), the deflection and internal forces of the column are obtained in an exact and closed form.

Buckling Analysis of Constrained Stepped Columns

Consider a constrained stepped column of n uniform segments as shown in Fig. 4.3.1. For the jth beam segment, its governing equation, Eq. (4.17), is cast into the state form

$$\frac{d}{dx}\eta_j(x) = F_j\eta_j(x), \quad 0 \le x \le L_j \tag{4.40}$$

where

$$\eta_j(x) = \begin{pmatrix} w_j(x) \\ w_j'(x) \\ w_j''(x) \\ w_j'''(x) \end{pmatrix}, \quad F_j = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\dfrac{k_j}{EI_j} & 0 & -\dfrac{P}{EI_j} & 0 \end{bmatrix} \tag{4.41}$$

The matching conditions at node j, Eqs. (4.18), have the equivalent state form

$$\eta_{j+1}(0) = T_j\eta_j(L_j) \tag{4.42}$$

where the connection matrix

$$T_j = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \dfrac{k_r}{EI_{j+1}} & \dfrac{EI_j}{EI_{j+1}} & 0 \\ -\dfrac{k_t}{EI_{j+1}} & 0 & 0 & \dfrac{EI_j}{EI_{j+1}} \end{bmatrix} \tag{4.43}$$

In addition, the boundary conditions of the column at node 0 and node n can be written as

$$[M_b]\{\eta_1(0)\} + [N_b]\{\eta_n(L)\} = 0. \tag{4.44}$$

The buckling problem of the constrained stepped column is defined by Eqs. (4.40), (4.42), and (4.44). The solution of Eq. (4.40) is written as

$$\eta_j(x) = e^{F_j x}\eta_j(0), \quad 0 \le x \le L_j. \tag{4.45}$$

By Eqs. (4.45) and (4.42)

$$\eta_n(L_n) = e^{F_n L_n}\eta_n(0) = e^{F_n L_n} T_{n-1}\eta_{n-1}(L_{n-1}).$$

Applying the above process recursively for $\eta_{n-1}, \eta_{n-2}, \ldots, \eta_2, \eta_1$, leads to

$$\eta_n(L_n) = \Phi(P)\eta_1(0) \tag{4.46}$$

where the transition matrix

$$\Phi(P) = e^{F_n L_n} T_{n-1} e^{F_{n-1} L_{n-1}} \cdots T_1 e^{F_1 L_1}. \tag{4.47}$$

Now, substitute Eq. (4.46) into the boundary conditions Eq. (4.44), to obtain the homogeneous equation

$$([M_b] + [N_b]\,\Phi(P))\,\eta_1(0) = 0. \tag{4.48}$$

The characteristic equation of the constrained stepped column then is

$$\det([M_b] + [N_b]\,\Phi(P)) = 0 \tag{4.49}$$

the roots of which, P_k, $k = 1, 2, \ldots$, are the buckling loads of the column. Once a buckling load P_k is obtained from Eq. (4.49), the associate mode shape can be obtained by solving Eq. (4.48) for $\eta_1(0)$. With the determined $\eta_1(0)$, the spatial distribution of modal deflection and internal forces of every segment is expressed by

$$\eta_1(x) = e^{F_1 x}\eta_1(0), \quad 0 \le x \le L_1 \tag{4.50a}$$

and

$$\eta_j(x) = e^{F_j x} T_{j-1} e^{F_{j-1} L_{j-1}} \cdots T_1 e^{F_1 L_1}\eta_1(0), \quad 0 \le x \le L_j \tag{4.50b}$$

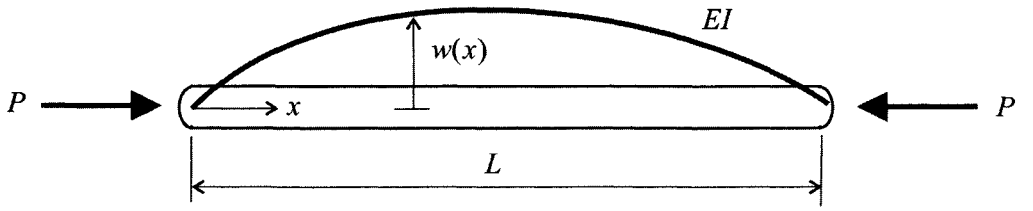
for $j = 2, 3, \ldots, n$.

In the above analysis, no approximation has been made; exact analytical buckling solutions for constrained stepped columns are obtained. This method is also applicable to nonuniform columns for highly accurate buckling solutions, as has been shown in Section 4.3.2.

4.5 Quick Solution Guide

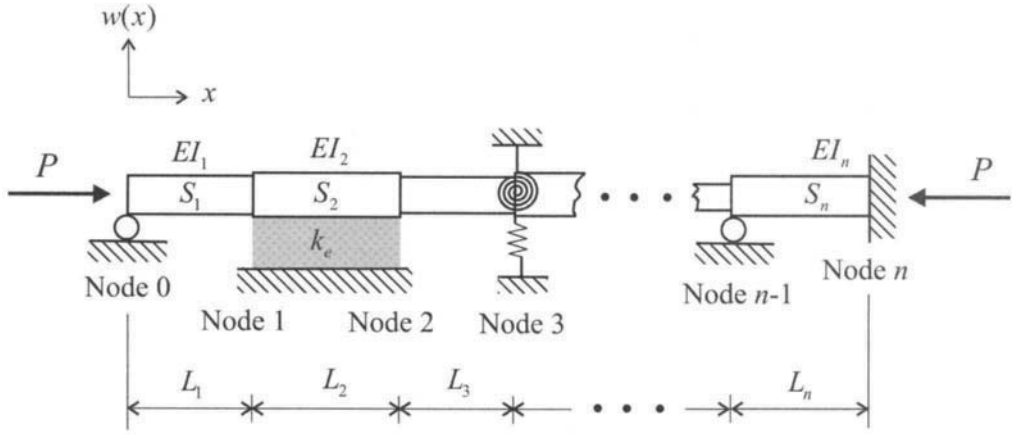
System Description

Uniform columns



$$EI\frac{d^4}{dx^4}w(x) + P\frac{d^2}{dx^2}w(x) = 0, \quad 0 \le x \le L$$

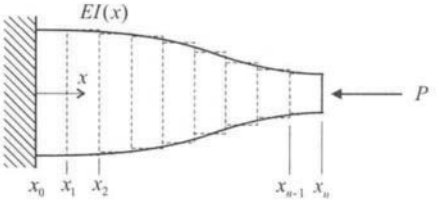
Constrained stepped columns



Nonuniform columns

$$\frac{d^2}{dx^2}\left(EI(x)\frac{d^2}{dx^2}w(x)\right) + P\frac{d^2}{dx^2}w(x) = 0$$

for $0 \le x \le L$



Problems to Solve

(a) Eigenvalue problem (buckling loads and modes shapes); and
(b) Response problem (deflection and internal forces of columns subject to transverse loads, eccentric axial loads, and initial bending)

MATLAB Functions

This chapter has a toolbox of MATLAB functions for buckling of columns; see the table below. Refer to the corresponding window or section for the utility of each function. The application of some main functions is summarized in the following pages.

Uniform Columns		
`setcolumn`	Compute the buckling loads and mode shapes of a uniform column	Window 2.1
`eigcolumn`	Determine buckling loads and mode shapes	Window 2.2
`plotmsh`	Plot and output buckling mode shapes	Window 2.3
`eccenload`	Compute the response of an eccentrically loaded column at a point versus the axial load	Window 2.4
`eccenload1`	Compute the spatial distribution of the response of an eccentrically loaded column	Window 2.5
`beamcolumn`	Compute the spatial distribution of the response of a beam-column subject to a transverse load	Window 2.6
`beamcolumn1`	Compute the response of a beam-column at a particular point versus the axial load	Window 2.7
`systinfo`	Display the information on the column and its eigensolutions	Section 4.2.2
Stepped and Nonuniform Columns		
`setcolumn2`	Compute the buckling loads and mode shapes of a constrained stepped column	Window 3.1
`plotmsh2`	Plot and output buckling mode shapes	Window 3.2
`nonucolumn`	Compute the buckling loads and mode shapes of a nonuniform column	Window 3.3
`systinfo2`	Display the information of a stepped or nonuniform column	Section 4.3
Utilities		
`TBdemo`	Show how the Toolbox works and what it can do	Section 4.1
`TBinfo`	Show the information of the functions in the Toolbox	Section 4.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 4.1

Solution Procedure

Buckling analysis of a column takes the following steps:

Uniform columns

To set up system and to obtain buckling loads, type

```
>> setcolumn(EI, L, BC_Spec)
```

To plot the kth buckling mode shape, type

```
>> plotmsh(k)
```

To obtain the response of a beam-column subject to a transverse load and an axial load `p`, type

```
>> beamcolumn(Load_Spec, p)
```

To obtain the deflection of a column subject to an axial load p with eccentricity e, type

```
>> eccenload1(e, p)
```

Constrained stepped columns

To obtain buckling loads and mode shapes, type

```
>> setcolumn2(Beam_Spec, BC_Spec, Constr_Spec)
```

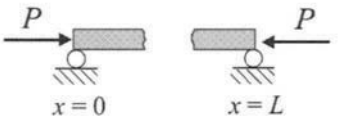
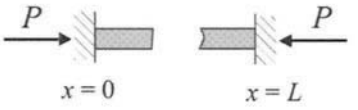
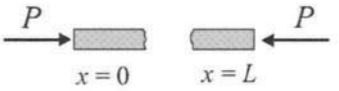
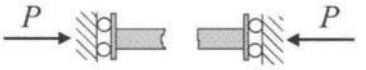
Nonuniform columns

To obtain buckling loads and mode shapes, type

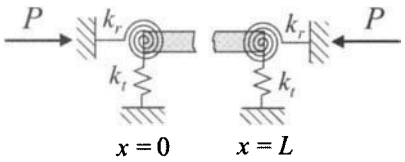
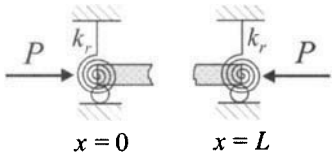
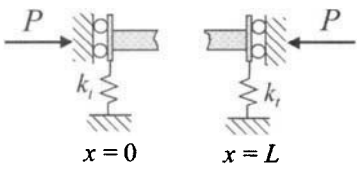
```
>> nonucolumn(EI_Spec, L, BC_Spec)
```

Here `BC_Spec` is a vector specifying the boundary conditions; `Load_Spec` is a vector specifying an external force; `Beam_Spec` and `Constr_Spec` are matrices specifying the beam segments and constraints of a stepped column, respectively; and `EI_Spec` is a vector or matrix specifying a nonuniform bending stiffness $EI(x)$. The formation of these vectors and matrices is given as follows.

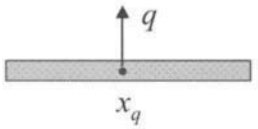
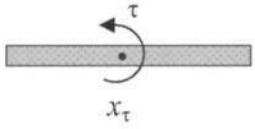
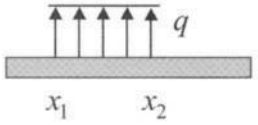
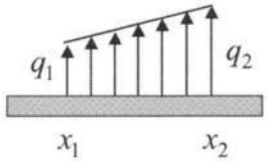
Specification of Boundary Conditions by Vector `BC_Spec`

Boundary Condition *(k_r - torsional spring coefficient; k_t - translational spring coefficient)*	`BC_Spec = [BC_Left BC_Right]`	
	Left Boundary ($x = 0$) `BC_Left`	*Right Boundary ($x = L$)* `BC_Right`
(B1) Pinned or hinged end 	[1]	[1]
(B2) Clamped or fixed end 	[2]	[2]
(B3) Free end 	[3]	[3]
(B4) Sliding or guided end 	[4]	[4]

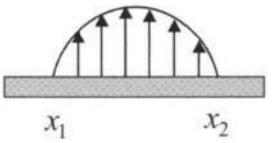
(continued)

Boundary Condition *(k_r - torsional spring coefficient;* *k_t - translational spring coefficient)*	**BC_Spec = [BC_Left BC_Right]**	
	Left Boundary (x = 0) BC_Left	*Right Boundary (x = L)* BC_Right
(B5) Elastic-support  	[5 k_r k_t]	[5 k_r k_t]
(B6) Hinged-elastic end  	[6 k_r]	[6 k_r]
(B7) Sliding-elastic end  	[7 k_t]	[7 k_t]

Specification of External Loads by Vector Load_Spec

(L1) Pointwise force  $f(x) = q\delta(x - x_q)$ **Load_Spec** = [0 x_q q]	(L2) Torque  $f(x) = -\tau \frac{d}{dx}\delta(x - x_\tau)$ **Load_Spec** = [−1 x_τ τ]
(D1) Uniform  $f(x) = q$ for $x_1 \le x \le x_2$ **Load_Spec** = [1 x_1 x_2 q]	(D2) Linear  $f(x) = q_1 + \frac{q_2 - q_1}{x_2 - x_1}(x - x_1)$ for $x_1 \le x \le x_2$ **Load_Spec** = [2 x_1 x_2 q_1 q_2]

(continued)

(D3) Quadratic	(D4) Sinusoidal
$f(x) = c_0 + c_1 x + c_2 x^2 \quad \text{for } x_1 \le x \le x_2$ **Load_Spec** = [3 x_1 x_2 c_0 c_1 c_2]	 $f(x) = A\sin(\omega x + \phi) \quad \text{for } x_1 \le x \le x_2$ **Load_Spec** = [4 x_1 x_2 A ω ϕ] (ϕ in radians)

Specification of Beam Segments of a Stepped Column by Vector `Beam_Spec`

$$\texttt{Beam\_Spec} = \begin{bmatrix} EI_1 & L_1 \\ EI_2 & L_2 \\ \vdots & \vdots \\ EI_n & L_n \end{bmatrix}$$

where the ith row of the matrix contains the bending stiffness and length of the ith segment. For beam segments constrained by elastic foundations or distributed springs,

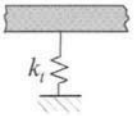
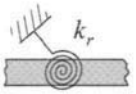
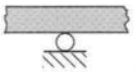
$$\texttt{Beam\_Spec} = \begin{bmatrix} EI_1 & L_1 & k_1 \\ EI_2 & L_2 & k_2 \\ \vdots & \vdots & \vdots \\ EI_n & L_n & k_n \end{bmatrix}$$

where k_j is the coefficient of elastic foundation or distributed spring imposed on the jth segment. If no distributed spring is imposed on the jth segment, k_j is set to zero

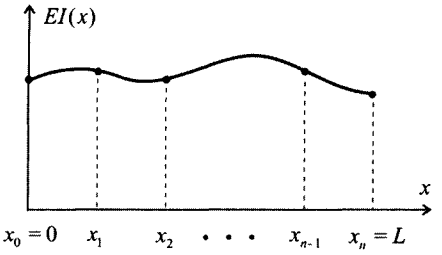
Specification of Constraints at Nodes by Vector `Constr_Spec`

$$\texttt{Constr\_Spec} = \begin{bmatrix} \texttt{Constr\_1} \\ \texttt{Constr\_2} \\ \vdots \\ \texttt{Constr\_Nc} \end{bmatrix}$$

The jth row vector of the matrix specifies the jth constraint, and is given in the table below, where `node_no` is the node number $(1, 2, \ldots, n-1)$. If the column has no constraints, set `Constr_Spec` = 0 or [ ].

Constraint Type	Constr_j
(C1) Translational spring 	[node_no 1 k_t]
(C2) Torsional spring 	[node_no 2 k_r]
(C3) Hinge 	[node_no 3]
(C4) Sliding constraint	[node_no 4]

Specification of Nonuniform Bending Stiffness by Vector EI_Spec

Distribution of Bending Stiffness	**EI_Spec**
Type 1 (linear or tapered) $EI(x) = EI_0\left(1 - \frac{x}{L}\right) + EI_L\frac{x}{L}$	EI_Spec = [1 EI_0 EI_L]
Type 2 (polynomial) $EI(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n$	EI_Spec = [2 c_0 c_1 c_2 $\cdots$ c_n]
Type 3 (sinusoidal) $EI(x) = a + b\sin(cx + \theta_0)$	EI_Spec = [3 a b c θ_0] θ_0 given in degrees
Type 4 $EI(x) = a\,[c_0 + c_1x + \cdots + c_nx^n]^b$ b = an arbitrary number	EI_Spec = [4 a b c_0 c_1 $\cdots$ c_n]
Type 5 (exponential) $EI(x) = a + be^{cx}$	EI_Spec = [5 a b c]
Type 6 (user-provided) 	EI_Spec = [6 $EI(x_0)$ $EI(x_1)\cdots$ $EI(x_n)$] Here $EI(x_0), EI(x_1), \ldots, EI(x_n)$ are the values of $EI(x)$ at the n+1 *equally-spaced* points $x_k = k\Delta$, $k = 0, 1, \ldots, n$, $\Delta = L/n$

(continued)

Distribution of Bending Stiffness	**EI_Spec**
Type 7 (user-provided) The distribution of $EI(x)$ is the same as shown in Type 6, except that the points $x_0, x_1, \ldots, x_n$ don't have to be equally spaced.	$\texttt{EI\_Spec} = \begin{bmatrix} EI(x_0) & EI(x_1) & \cdots & EI(x_n) \\ x_0 & x_1 & \cdots & x_n \end{bmatrix}$ Here $EI(x_0), EI(x_1), \ldots, EI(x_n)$ are the values of $EI(x)$ at the n+1 points x_k, $k = 0, 1, \ldots, n$, with $x_0 = 0$ and $x_n = L$.

4.6 References

1. Popov E P 1998 *Engineering Mechanics of Solids*, 2nd ed., Prentice-Hall: Upper Saddle River, New Jersey.
2. Shames I H, Pitarresi J M 2000 *Introduction to Solid Mechanics*, 3rd ed., Prentice-Hall: Upper Saddle River, New Jersey.
3. Timoshenko S P, Gere J M 1961 *Theory of Elastic Stability*, 2nd ed., McGraw-Hill: New York, Toronto, London.
4. Yang B, Park D H 2003 "Exact Buckling Analysis of Constrained Stepped Columns," *International Journal of Structural Stability and Dynamics*, Vol. No. 2, pp. 143–167.

This Page Intentionally Left Blank

5 Stress Analysis in Two-Dimensional Problems

Inside
- Getting Started
- Plane Stress and Strain
- Quick Solution Guide
- References

5.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, and a set (toolbox) of MATLAB functions for stress analysis in two-dimensional problems.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

For a quick solution, the reader may refer directly to the Quick Solution Guide (Section 5.3), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To understand better how the Toolbox works, the reader is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 5.4. In addition, the reader may refer to Chapter 15 for elasticity theory.

5.2 Plane Stress and Strain

5.2.1 In-Plane Stresses

Sign Convention and Units In Fig. 5.2.1, the state of plane stress at a point of elastic body is represented by two normal stress components (σ_x, σ_y) and one shear stress component (τ_{xy}), which act on four sides of an element in the coordinates x, y. The figure also shows the sign convention for the stress components: (i) positive normal stress acts outwards from all the sides of the element; (ii) positive shear stress acts upward on the righthand side, and acts in the positive x-direction on the top side.

The dimension of stress is force per unit area. The units of stress in the International Standard (SI system) and the U.S. Customary are given in Table 5.2.1.

Stress Transformation The state of plane stress given in Fig. 5.2.1 can also be described on another element with a different orientation; see in Fig. 5.2.2 where angle θ is measured from the positive x axis to the positive x' axis, counterclockwise. In other words, a state of stress can be defined by either of the two sets of stress components: (σ_x, σ_y, τ_{xy}) along the x, y

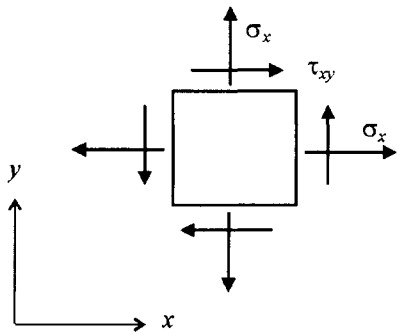


FIGURE 5.2.1 Stress components of an element.

TABLE 5.2.1 Units of Stress

International Standard	*U.S. Customary*
1 Pa = 1 N/m^2	1 psi = 1 lb/in^2
1 kPa = 10^3 Pa	1 ksi = 10^3 lb/in^2
1 MPa = 10^6 Pa	
1 GPa = 10^9 Pa	
Conversion	1 psi = 6.89476 kPa; 1 kPa = 0.14504 psi 1 ksi = 6.89476 MPa; 1 MPa = 0.14504 ksi

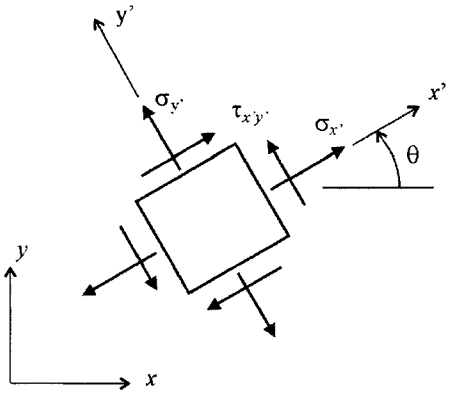


FIGURE 5.2.2 Stress components of an element of angle θ.

axes and $(\sigma_{x'}, \sigma_{y'}, \tau_{x'y'})$ along the $x'y'$ axes. The transformation of stress components from the coordinates x, y to the coordinates $x'y'$ is given by

$$\begin{aligned}
\sigma_{x'} &= \frac{\sigma_x + \sigma_y}{2} + \frac{\sigma_x - \sigma_y}{2}\cos 2\theta + \tau_{xy}\sin 2\theta \\
\sigma_{y'} &= \frac{\sigma_x + \sigma_y}{2} - \frac{\sigma_x - \sigma_y}{2}\cos 2\theta - \tau_{xy}\sin 2\theta \\
\tau_{x'y'} &= -\frac{\sigma_x - \sigma_y}{2}\sin 2\theta + \tau_{xy}\cos 2\theta.
\end{aligned} \tag{5.1}$$

The Toolbox of this chapter has a function `stressTF` for the above stress transformation; see Window 2.1.

Window 2.1. Function stressTF

MATLAB Function: **stressTF**

Purpose:
To transform stress components from an element to another of different orientation.

Synopsis:

```
stressTF(s, theta)
y = stressTF(s, theta)
```

Description:
`stressTF(s, theta)` transforms stress components $\sigma_x, \sigma_y, \tau_{xy}$ oriented along the x, y axes to $\sigma_{x'}, \sigma_{y'}, \tau_{x'y'}$ oriented along the x', y' axes (see Fig. 5.2.2). Here `s` is a row vector of stress components, i.e., `s` = $[\sigma_x \ \ \sigma_y \ \ \tau_{xy}]$; `theta` is an angle that is measured from the positive x axis to the positive x' axis, counterclockwise. The value of `theta` is assigned in degrees.
`y = stressTF(s, theta)` returns the computed stress components oriented along the x', y' axes in a row vector `y` = $[\sigma_{x'} \ \ \sigma_{y'} \ \ \tau_{x'y'}]$.

EXAMPLE 2.1

The state of stress at a point is defined by $\sigma_x = -70$ MPa, $\sigma_y = 55$ MPa, $\tau_{xy} = -15$ MPa. Consider another element in x', y' with the angle from the positive x axis to the positive x' axis being $\theta = 60°$. The stress components in the new coordinates x', y' are determined by typing the following in the MATLAB command window:

```
>> s=[-70 55 -15];
>> stressTF(s, 60)
```

where >> is the prompt in the MATLAB command window. This yields

```
Plane-Stress Transformation
---------------------------
Given state of plane stress in the coordinate system oxy:
   normal stress Sx = -70
   normal stress Sy = 55
   shear stress Txy = -15

Stress components in another coordinate system ox' y' with an
angle
from x axis to x' axis (counterclockwise) being 60 degrees:
   normal stress Sx' = 10.7596
   normal stress Sy' = -25.7596
   shear stress Tx'y' = 61.6266
```

So, the stress components in x', y' are $\sigma_{x'} = 10.8$ MPa, $\sigma_y = -25.8$ MPa, and $\tau_{xy} = 61.6$ MPa.

Principal Stresses The principal stresses (maximum and minimum normal stresses) and the orientation θ of the element are obtained by differentiating σ_x and σ_y in Eq. (5.1) with respect to θ, which yields the following results:

Principal stresses

$$\sigma_{1,2} = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2} \tag{5.2}$$

Orientation of the planes of principal stresses

$$\tan 2\theta_p = \frac{\tau_{xy}}{(\sigma_x - \sigma_y)/2} \tag{5.3}$$

Equation (5.3) has two roots θ_{p1} and θ_{p2} for σ_1 and σ_2, respectively; see Fig. 5.2.3. Note that $\theta_{p2} = \theta_{p1} + 90°$. Furthermore, no shear stress acts on the principal planes, i.e., $\tau_{x'y'} = 0$ on the planes of orientation θ_{p1} and θ_{p2}.

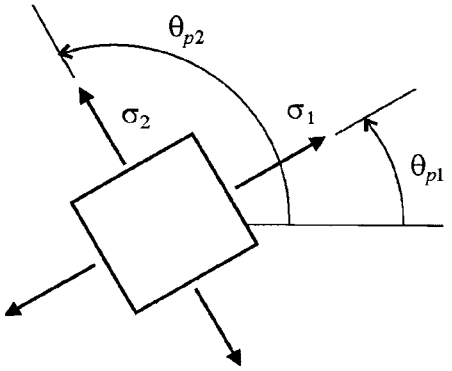


FIGURE 5.2.3 Principal stresses.

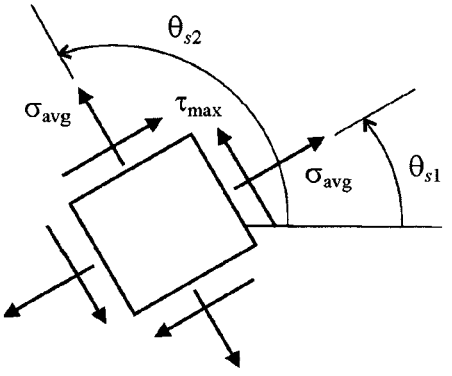


FIGURE 5.2.4 Maximum shear stress and average normal stress.

Maximum In-Plane Shear Stress The maximum shear stress and corresponding planes are determined by differentiating τ_{xy} with respect to θ, which yields the following results:

Maximum in-plane shear stress

$$\tau_{\max} = \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2} \tag{5.4}$$

Orientation of the planes of maximum shear stress

$$\tan 2\theta_s = \frac{-(\sigma_x - \sigma_y)/2}{\tau_{xy}} \tag{5.5}$$

Equation (5.5) has two roots θ_{s1} and θ_{s2}; see Fig. 5.2.4. On each of the planes of maximum shear stress, there acts a normal stress (called *average normal stress*)

$$\sigma_{avg} = \frac{\sigma_x + \sigma_y}{2}. \tag{5.6}$$

A comparison of Eqs. (5.3) and (5.5) shows that the orientation θ_p of the principal planes and the orientation θ_s of the planes of maximum shear stress are 45° apart.

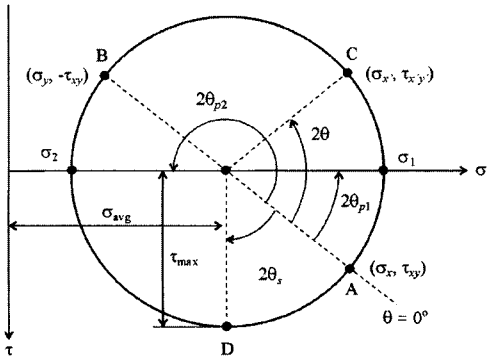


FIGURE 5.2.5 Mohr's circle of stress.

Mohr's Circle for Plane Stress In-plane stress transformation, principal stresses, and maximum shear stress can be presented graphically in Mohr's circle; see Fig. 5.2.5 where the coordinates σ, τ represent normal and shear stresses, respectively. The equation of the circle is

$$(\sigma_{x'} - \sigma_{avg})^2 + \tau_{x'y'}^2 = R^2 \tag{5.7}$$

The radius and center of the circle are

$$R = \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2}$$

and $(\sigma_{avg}, 0)$, respectively. There are two rules about Mohr's circle: (a) the positive τ axis is downward; and (b) double angles $2\theta, 2\theta_p, 2\theta_s$ are measured in the counterclockwise direction.

The representation of stress transformation, principal stresses, and maximum shear stress in Mohr's circle is as follows:

- A state of stress is represented by point A ($\sigma = \sigma_x, \tau = \tau_{xy}$) and point B ($\sigma = \sigma_y, \tau = -\tau_{xy}$) on the circle. Here points A and B are 180° apart.
- The stresses acting on an element of orientation θ are represented by point C ($\sigma = \sigma_{x'}, \tau = \tau_{x'y'}$) and another point ($\sigma = \sigma_{y'}, \tau = -\tau_{x'y'}$) on the circle that is 180° from point C (not shown).
- The principal stresses σ_1 and σ_2 are represented by the intersections of the circle and the σ axis; the corresponding angles $2\theta_{p1}$ and $2\theta_{p2}$ are measured from point A. Shear stress τ at the intersections is zero.
- The maximum shear stress $\tau_{\max}$ and the average normal stress σ_{avg} are represented by point D, which is $-2\theta_s$ away from point A.

The Toolbox of this chapter provides a function `pristress` for determining principal stresses and maximum shear stress and for plotting Mohr's circle; see Window 2.2.

Window 2.2. Function `pristress`

MATLAB Function: **`pristress`**

Purpose:
To compute principal stresses and maximum in-plane shear stress and to plot Mohr's circle for plane stress.

Window 2.2. Function pristress (*Continued*)

Synopsis:

```
pristress(s)
y = pristress(s)
```

Description:

`pristress(s)` carries out two tasks for a given state of in-plane stress expressed by vector `s` = $[\sigma_x \quad \sigma_y \quad \tau_{xy}]$. First, it computes principal stresses, maximum in-plane shear stress, and corresponding orientation angles. Second, it plots the Mohr's circle for the given state of stress.

`y = pristress(s)` returns the computed results in a matrix

$$\mathtt{y} = \begin{bmatrix} \sigma_1 & \theta_{p1} \\ \sigma_2 & \theta_{p2} \\ \tau_{\max} & \theta_{s1} \\ \sigma_{avg} & \theta_{s2} \end{bmatrix}$$

Here, the first column of the matrix contains principal stresses, maximum in-plane shear stress, and average normal stress; the second column contains the corresponding orientation angles (in degrees).

EXAMPLE 2.2

Consider the state of stress given in Example 2.1: $\sigma_x = -70$ MPa, $\sigma_y = 55$ MPa, $\tau_{xy} = -15$ MPa. The principal stresses and maximum in-plane shear stress are obtained by

```
>> pristress([-70e6, 55e6, -15e6])
```

which yields the following results

```
Principal Stresses and Maximum In-Plane Shear Stress
-----------------------------------------------------
Given stress components:
   Sx = -70000000
   Sy = 55000000
   Txy= -15000000
Principal stresses:
   S1 = 56774800.6609 with ang_p1 = -83.2521 deg
   S2 = -71774800.6609 with ang_p2 = 6.7479 deg
Maximum in-plane shear stress:
   Tmax = 64274800.6609 with ang_T = 51.7479 deg and -38.2521 deg
   On the planes of Tmax, average normal stress Savg = (Sx + Sy)/2
   = -7500000
Mohr's circle (see figure):
   * angles are measured counterclockwise
```

and the Mohr's circle in Fig. 5.2.6.

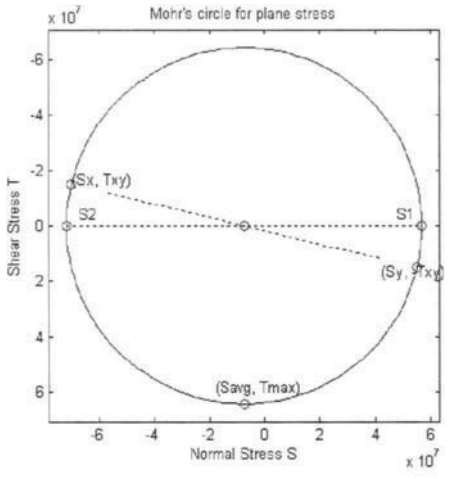


FIGURE 5.2.6

5.2.2 In-Plane Strains

A plane-strained element is subject to two components of normal strain ($\varepsilon_x, \varepsilon_y$) and one component of shear strain (γ_{xy}); see Fig. 5.2.7 where normal strains are due to changes in length while shear strain is due to the relative rotation of two adjacent sides of the element. Normal strains $\varepsilon_x, \varepsilon_y$ are positive if they cause elongation along the x and y axes, respectively. Shear strain γ_{xy} is positive if angle α of the element in Fig. 5.2.7 is less than 90°. Strain components are length ratios, and thus are dimensionless.

Due to Poisson's effect, a state of plane strain is not the same as a state of plane stress. Also, refer to Chapter 15 for stress and strain in two- and three-dimensions.

Strain Transformation Similar to stress transformation, the state of plane strain at a point can be expressed by components in two elements of different orientations: $\varepsilon_x, \varepsilon_y, \gamma_{xy}$ in the coordinates x, y, and $\varepsilon_{x'}, \varepsilon_{y'}, \gamma_{x'y'}$ in the coordinates x', y'; see Fig. 5.2.8 where the angle

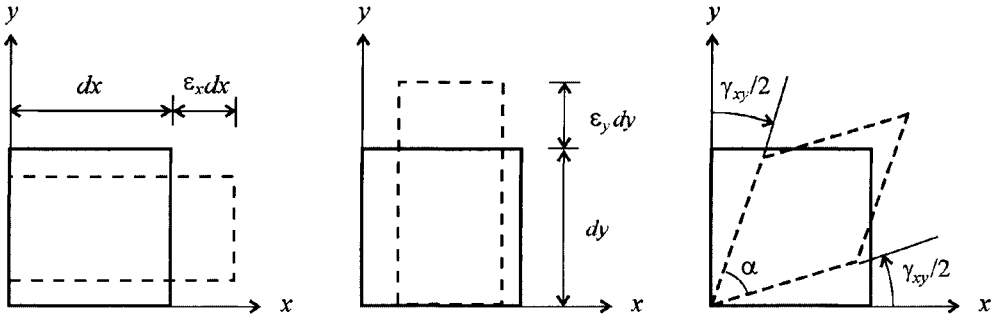


FIGURE 5.2.7 Strain components.

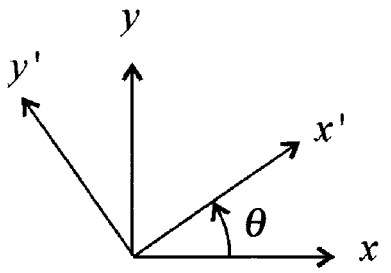


FIGURE 5.2.8

between the positive x and x' axes (measured counterclockwise) is θ. The strain components $\varepsilon_{x'}, \varepsilon_{y'}, \gamma_{x'y'}$ can be expressed in terms of $\varepsilon_x, \varepsilon_y, \gamma_{xy}$ by

$$\begin{aligned}
\varepsilon_{x'} &= \frac{\varepsilon_x + \varepsilon_y}{2} + \frac{\varepsilon_x - \varepsilon_y}{2}\cos 2\theta + \frac{\gamma_{xy}}{2}\sin 2\theta \\
\varepsilon_{y'} &= \frac{\varepsilon_x + \varepsilon_y}{2} - \frac{\varepsilon_x - \varepsilon_y}{2}\cos 2\theta - \frac{\gamma_{xy}}{2}\sin 2\theta \\
\frac{\gamma_{x'y'}}{2} &= -\frac{\varepsilon_x - \varepsilon_y}{2}\sin 2\theta + \frac{\gamma_{xy}}{2}\cos 2\theta.
\end{aligned} \tag{5.8}$$

The Toolbox of this chapter has a function `strainTF` for the above strain transformation; see Window 2.3.

Window 2.3. Function `strainTF`

MATLAB Function: **`strainTF`**

Purpose:
To transform strain components from an element to another of different orientation.

Synopsis:

```
strainTF(s, theta)
y = strainTF(s, theta)
```

Description:
`strainTF(s, theta)` transforms stress components (ε_x, ε_y, γ_{xy}) oriented along the x, y axes to those ($\varepsilon_{x'}$, $\varepsilon_{y'}$, $\gamma_{x'y'}$) oriented along the x', y' axes. Here `s` is a row vector of given strain components, i.e., `s` = [ε_x ε_y γ_{xy}]; and `theta` is an angle that is measured from the positive x axis to the positive x' axis counterclockwise. The value of `theta` is assigned in degrees.
`y = strainTF(s, theta)` returns the computed stress components oriented along the x', y' axes in a row vector `y` = [$\varepsilon_{x'}$ $\varepsilon_{y'}$ $\gamma_{x'y'}$].

EXAMPLE 2.3

A state of plane strain is expressed by $\varepsilon_x = 40 \times 10^{-4}$, $\varepsilon_x = -25 \times 10^{-4}$, and $\gamma_{xy} = 20 \times 10^{-4}$ along the x, y axes. Consider another element in x', y' with the angle between the positive x and x' axes being $\theta = 30°$ counterclockwise. To determine the strain components in the new coordinates x', y', type the following MATLAB commands:

```
>> s=[40e-4 -25e-4 20e-4];
>> strainTF(s,30)
```

which yield

```
Plane-Strain Transformation
---------------------------
Given state of plane strain in the coordinate system oxy:
   normal strain Ex = 0.004
   normal strain Ey = -0.0025
   shear strain Lxy= 0.002

Strain components in another coordinate system ox'y' with an angle
from x axis to x' axis (counterclockwise) being 30 degrees:
   normal strain Ex' = 0.003241
   normal strain Ey' = -0.001741
   shear strain Lx'y' = -0.0046292
```

Principal Strains The principal strains (maximum and minimum normal strains), which are obtained by differentiating ε_x, ε_y with respect to θ, are

$$\varepsilon_{1,2} = \frac{\varepsilon_x + \varepsilon_y}{2} \pm \sqrt{\left(\frac{\varepsilon_x - \varepsilon_y}{2}\right)^2 + \left(\frac{\gamma_{xy}}{2}\right)^2}. \tag{5.9}$$

The orientation of the planes of the principal strains is determined by

$$\tan 2\theta_p = \frac{\gamma_{xy}}{\varepsilon_x - \varepsilon_y} \tag{5.10}$$

which has two roots θ_{p1} and θ_{p2}, for ε_1 and ε_2, respectively. The θ_{p1} and θ_{p2} are 90° apart. Furthermore, no shear strain acts on the planes of the principal strains.

If the material of the elastic body is isotropic, the axes of the principal strains coincide with those of the principal stresses.

Maximum In-Plane Shear Strain The maximum in-plane shear strain and the associate orientation are determined by

$$\frac{\gamma_{\max}}{2} = \sqrt{\left(\frac{\varepsilon_x - \varepsilon_y}{2}\right)^2 + \left(\frac{\gamma_{xy}}{2}\right)^2} \tag{5.11}$$

and

$$\tan 2\theta_s = -\frac{\varepsilon_x - \varepsilon_y}{\gamma_{xy}}. \tag{5.12}$$

Equation (5.12) has two roots, θ_{s1} and θ_{s2}. The θ_p and θ_s are 45° apart. On each of the planes of maximum in-plane shear stress, a normal strain (called *average normal strain*) is

$$\varepsilon_{avg} = \frac{\varepsilon_x + \varepsilon_y}{2} \tag{5.13}$$

Mohr's Circle for Plane Strain In-plane strain transformation, principal strain and maximum shear strain can be presented graphically in Mohr's circle; see Fig. 5.2.9 where the horizontal axis (ε) represents normal strain and the downward vertical axis ($\gamma/2$) represents half of shear strain, and angles 2θ, $2\theta_p$, $2\theta_s$ are measured in the counterclockwise direction. The equation for Mohr's circle is

$$\left(\varepsilon_{x'} - \varepsilon_{avg}\right)^2 + \left(\frac{\gamma_{x'y'}}{2}\right)^2 = R^2 \tag{5.14}$$

The radius and center of the circle are

$$R = \sqrt{\left(\frac{\varepsilon_x - \varepsilon_y}{2}\right)^2 + \left(\frac{\gamma_{xy}}{2}\right)^2}$$

and (ε_{avg}, 0), respectively.

Strain transformation, principal stresses, and maximum shear stress are expressed in Mohr's circle as follows:

- A state of plane strain is represented by point A ($\varepsilon = \varepsilon_x$, $\gamma/2 = \gamma_{xy}/2$) and point B ($\varepsilon = \varepsilon_y$, $\gamma/2 = -\gamma_{xy}/2$). Points A and B are 180° apart.
- The strains acting on an element of orientation θ are represented by point C ($\varepsilon = \varepsilon_{x'}$, $\gamma/2 = \gamma_{x'y'}/2$) and another point ($\varepsilon = \varepsilon_{y'}$, $\gamma/2 = -\gamma_{x'y'}/2$, not shown).
- The principal strains ε_1 and ε_2 are represented by the intersections of the circle and the ε axis; the corresponding angles $2\theta_{p1}$ and $2\theta_{p2}$ are measured from point A.
- The maximum shear strain $\gamma_{\max}$ and the average normal strain ε_{avg} are represented by point D, which is $-2\theta_s$ away from point A.

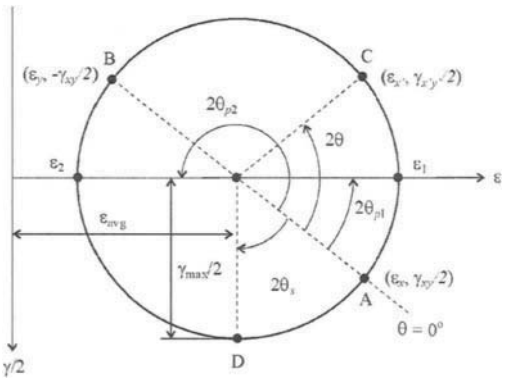


FIGURE 5.2.9 Mohr's circle of strain.

This Toolbox provides a function `pristrain` for determining maximum principal strains and maximum shear strain, and for plotting Mohr's circle; see Window 2.4.

Window 2.4. Function `pristrain`

MATLAB Function: **`pristrain`**

Purpose:
To compute principal strains and maximum in-plane shear strain and to plot Mohr's circle for plane strain.

Synopsis:

```
pristrain(s)
y = pristrain(s)
```

Description:
`pristrain(s)` carries out two tasks for a given state of in-plane strain expressed by vector `s` = $[\varepsilon_x\ \varepsilon_y\ \gamma_{xy}]$. First, it computes principal strains, maximum in-plane shear strain, and corresponding orientation angles. Second, it plots the Mohr's circle for the given state of strain.
`y = pristrain(s)` returns the computed results in a two-row matrix

$$\mathtt{y} = \begin{bmatrix} \varepsilon_1 & \theta_{p1} \\ \varepsilon_2 & \theta_{p2} \\ \gamma_{\max} & \theta_{s1} \\ \varepsilon_{avg} & \theta_{s2} \end{bmatrix}$$

Here, the first column of the matrix contains principal strains, maximum in-plane shear strain, and average normal strain; the second column contains the corresponding orientation angles (in degrees).

EXAMPLE 2.4

For the state of plane strain given in Example 2.3, the principal strains and maximum in-plane shear strain are obtained by

```
>> pristrain([40e-4 -25e-4 20e-4])
```

which plots Mohr's circle plotted in Fig. 5.2.6 and gives the following information;

```
Principal Strains and Maximum In-Plane Shear Strain
---------------------------------------------------
Given stress components:
   normal strain Ex = 0.004
   normal strain Ey = -0.0025
   shear strain Lxy= 0.002

 Principal strains:
   E1 = 0.0041504 with ang_p1 = 8.5514 deg
   E2 = -0.0026504 with ang_p2 = -81.4486 deg
```

```
Maximum in-plane shear strain:
   Lmax = 0.0068007 with ang_L = -36.4486 deg and 53.5514 deg
   On the plane of Lmax, average normal strain Eavg = (Ex + Ey)/2
   = 0.00075

Mohr's circle (see figure):
   * angles are measured counterclockwise
```

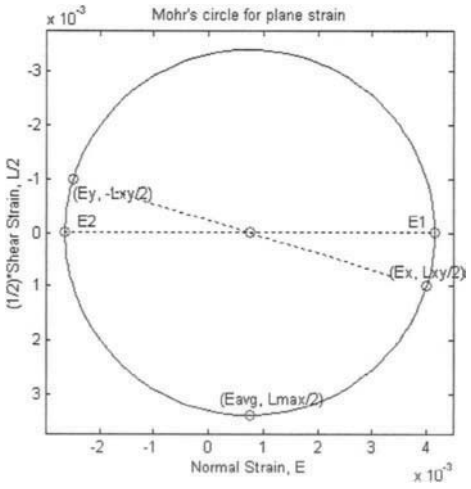


FIGURE 5.2.10

5.2.3 Strain Rosette

In many cases of engineering practice, the normal strains at a point on the surface of a body can be determined through use of a cluster of three electrical-resistance strain gauges. These strain gauges are usually arranged in a special pattern, which is referred to as a *strain rosette*. In general, the axes of the three gauges are arranged at the angles θ_a, θ_b, and θ_c; see Fig. 5.2.11. Each strain gauge can only measure the normal strain along its axis. For instance, strain gauge a in Fig. 5.2.11 measures normal strain ε_a along its axis, which makes an angle θ_a with the x axis. Once the readings ε_a, ε_b, ε_c from the strain gauges are taken, the strain components ε_x, ε_y, γ_{xy} at the point can be computed as follows. By Eq. (5.8), the strain readings are expressed by

$$\begin{aligned}
\varepsilon_a &= \varepsilon_x \cos^2\theta_a + \varepsilon_y \sin^2\theta_a + \gamma_{xy} \sin\theta_a \cos\theta_a \\
\varepsilon_b &= \varepsilon_x \cos^2\theta_b + \varepsilon_y \sin^2\theta_b + \gamma_{xy} \sin\theta_b \cos\theta_b \\
\varepsilon_c &= \varepsilon_x \cos^2\theta_c + \varepsilon_y \sin^2\theta_c + \gamma_{xy} \sin\theta_c \cos\theta_c
\end{aligned} \tag{5.15}$$

The strain components are then obtained as

$$\begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix} = \begin{bmatrix} \cos^2\theta_a & \sin^2\theta_a & \sin\theta_a \cos\theta_a \\ \cos^2\theta_b & \sin^2\theta_b & \sin\theta_b \cos\theta_b \\ \cos^2\theta_c & \sin^2\theta_c & \sin\theta_c \cos\theta_c \end{bmatrix}^{-1} \begin{pmatrix} \varepsilon_a \\ \varepsilon_b \\ \varepsilon_c \end{pmatrix} \tag{5.16}$$

Table 5.2.2 lists two commonly used strain rosettes: 45° strain rosette and 60° strain rosette.

The Toolbox of this chapter provides a function `strgauges` that computes strain components based on the strain gauge readings of a strain rosette; see Window 2.5.

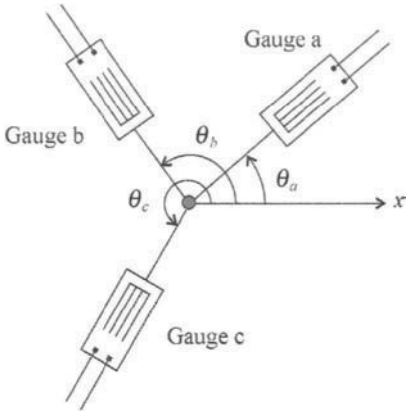


FIGURE 5.2.11 A strain rosette.

TABLE 5.2.2

Strain Gauge Layout		*Strain Computation Formulas*
45° strain rosette	$\theta_a = 0°, \theta_b = 45°, \theta_c = 90°$	$\varepsilon_x = \varepsilon_a, \quad \varepsilon_y = \varepsilon_c$ $\gamma_{xy} = 2\varepsilon_b - (\varepsilon_a + \varepsilon_c)$
60° strain rosette	$\theta_a = 0°, \theta_b = 60°, \theta_c = 120°$	$\varepsilon_x = \varepsilon_a, \quad \varepsilon_y = \frac{1}{3}(2\varepsilon_b + 2\varepsilon_c - \varepsilon_a)$ $\gamma_{xy} = \frac{2}{\sqrt{3}}(\varepsilon_b - \varepsilon_c)$

Window 2.5. Function strgauges

MATLAB Function: **strgauges**

Purpose:
To determine strain components from the readings of a strain rosette.

Synopsis:

```
strgauges(SG_Readings)
y = strgauges(SG_Readings)
```

Description:
`strgauges(SG_Readings)` computes the strain components ε_x, ε_y, γ_{xy} at a point based on the readings ε_a, ε_b, ε_c from three strain gauges a, b, and c arranged at angles θ_a, θ_b, and θ_c, respectively. Here `SG_Readings` is a matrix of the form

$$\texttt{SG\_Readings} = \begin{bmatrix} \varepsilon_a & \theta_a \\ \varepsilon_b & \theta_b \\ \varepsilon_c & \theta_c \end{bmatrix}$$

with the angles in degrees. `strgauges` also shows the principal strains and maximum shear strain at the point.

`y = strgauges(SG_Readings)` returns the computed strain components in a vector

$$\texttt{y} = [\varepsilon_x \;\; \varepsilon_y \;\; \gamma_{xy}].$$

EXAMPLE 2.5

The state of strain at a point on a machine part is measured by a 45° strain rosette ($\theta_a = 0°$, $\theta_b = 45°$, $\theta_c = 90°$). The strain gauge readings are $\varepsilon_a = 60\ \mu$m/m, $\varepsilon_b = 125\ \mu$m/m, and $\varepsilon_c = 100\ \mu$m/m, where $1\ \mu\text{m} = 10^{-6}$m. The strain components ε_x, ε_y, γ_{xy} and the principal strains at the point are computed by

```
>> SG_Readings =[60e-6 0; 125e-6 45; 100e-6 90];
>> strgauges(SG_Readings)
```

which yields the following results

```
Strain Components by Strain Rosette
-----------------------------------
Given strain readings
   Strain Gauge A: angle = 0 deg, strain reading = 6e-005
   Strain Gauge B: angle = 45 deg, strain reading = 0.000125
   Strain Gauge C: angle = 90 deg, strain reading = 0.0001

Computed strain components:
   normal strain Ex = 6e-005
   normal strain Ey = 0.0001
   shear strain Lxy = 9e-005

Principal strains:
   E1 = 0.00012924 with ang_p1 = 56.9812 deg
   E2 = 3.0756e-005 with ang_p2 = -33.0188 deg

Maximum in-plane shear strain:
   Lmax = 9.8489e-005 with ang_L = 11.9812deg and 101.9812 deg
   On the plane of Lmax, average normal strain Eavg = (Ex + Ey)/2
   = 8e-005
```

5.2.4 Hooke's Law

The general Hooke's law for isotropic linearly elastic materials reads (Reference 2)

$$\begin{aligned}
\varepsilon_x &= \frac{1}{E}\left[\sigma_x - \mu(\sigma_y + \sigma_z)\right] \\
\varepsilon_y &= \frac{1}{E}\left[\sigma_y - \mu(\sigma_x + \sigma_z)\right] \\
\varepsilon_z &= \frac{1}{E}\left[\sigma_z - \mu(\sigma_x + \sigma_y)\right] \\
\gamma_{xy} &= \frac{1}{G}\tau_{xy}, \quad \gamma_{yz} = \frac{1}{G}\tau_{yz}, \quad \gamma_{xz} = \frac{1}{G}\tau_{xz}
\end{aligned} \tag{5.17}$$

Here (ε_x, ε_y, ε_z, γ_{xy}, γ_{yz}, γ_{xz}) and (σ_x, σ_y, σ_z, τ_{xy}, τ_{yz}, τ_{xz}) are strain and stress components defined in three dimensions, E is Young's modulus, μ is Poisson's ratio, and G is shear modulus of the form

$$G = \frac{E}{2(1+\mu)} \tag{5.18}$$

The units of E and G are the same as stresses.

If the stress components σ_z, τ_{yz}, τ_{xz} are negligible, the generalized Hooke's law reduces one for the x-y plane:

$$\begin{aligned} \varepsilon_x &= \frac{1}{E}\left[\sigma_x - \mu\sigma_y\right] \\ \varepsilon_y &= \frac{1}{E}\left[\sigma_y - \mu\sigma_x\right] \\ \gamma_{xy} &= \frac{1}{G}\tau_{xy} \end{aligned} \tag{5.19}$$

On the other hand, if the strain components ε_z, γ_{yz}, γ_{xz} are negligible, the in-plane stress components can be expressed as

$$\begin{aligned} \sigma_x &= \frac{E}{1-\mu^2}\left[\varepsilon_x + \mu\varepsilon_y\right] \\ \sigma_y &= \frac{E}{1-\mu^2}\left[\varepsilon_y + \mu\varepsilon_x\right] \\ \tau_{xy} &= G\gamma_{xy} \end{aligned} \tag{5.20}$$

The Toolbox of this chapter has two functions, `hooks2e`, and `hooke2s`, for computing in-plane stresses and strains according to Eqs. (5.19) and (5.20); see Window 2.6.

Window 2.6. Functions hooks2e and hooke2s

MATLAB Functions: **hooks2e, hooke2s**

Purpose:
To obtain strain or stress components based on Hooke's law.

Synopsis:

```
hooks2e(stresses, E, Mu)
y = hooks2e(stresses, E, Mu)
hooke2s(strains, E, Mu)
y = hooke2s(strains, E, Mu)
```

Description:
`hooks2e(stresses, E, Mu)` computes strain components based on Eq. (5.19), where `stresses` is a vector of given stress components, i.e., `stresses` = $[\sigma_x \ \sigma_y \ \tau_{xy}]$; `E` is Young's modulus, and `Mu` is Poisson's ratio. In addition, `hooks2e` displays principal strains and maximum in-plane shear strain.

`y = hooks2e(stresses, E, Mu)` returns the computed strain components in a vector `y` = $[\varepsilon_x \ \varepsilon_y \ \gamma_{xy}]$.

`hooke2s(strains, E, Mu)` computes stress components based on Eq. (5.20), where `strains` is a vector of given strain components, i.e., `strains` = $[\varepsilon_x \ \varepsilon_y \ \gamma_{xy}]$. Also, `hooke2s` displays principal stresses and maximum in-plane shear stress.

`y = hooke2s(strains, E, Mu)` returns the computed stress components in a vector `y` = $[\sigma_x \ \sigma_y \ \tau_{xy}]$.

EXAMPLE 2.6

The state of strain at a point is described by $\varepsilon_x = 60 \times 10^{-5}$, $\varepsilon_y = -76 \times 10^{-5}$, and $\gamma_{xy} = -120 \times 10^{-5}$. Assume that $E = 175$ GPa and $\mu = 0.32$. The stress components and principal stresses are computed by

```
>> hooke2s([60e-5 -76e-5 -120e-5], 175e9, 0.32)
```

which gives

```
Stresses by Hooke's Law in Two Dimensions
-----------------------------------------
Given strains and elastic coefficients:
   normal strain Ex = 0.0006
   normal strain Ey = -0.00076
   shear strain Lxy = -0.0012
   Young's modulus E = 1.750000e+011
   Poisson's ratio mu = 0.32

Stresses:
   normal stress Sx = 69563279.8574
   normal stress Sy = -110739750.4456
   shear stress Txy = -79545454.5455

Principal stresses:
   S1 = 99639777.3442 with ang_p1 = -20.7118 deg
   S2 = -140816247.9324 with ang_p2 = 69.2882 deg

Maximum in-plane shear stress:
   Tmax = 120228012.6383 with ang_T = 114.2882 deg and 24.2882 deg
   On the planes of Tmax, average normal stress Savg = (Sx + Sy)/2
   = -20588235.2941
```

5.2.5 Criteria of Failure

Material failure is one important issue in design of machines and structures. Under high-stress loadings, a ductile material fails by yielding and a brittle material fails by fracture. Criteria for predicting an upper limit on the state of stress that defines a material's failure have been developed. In this section, three commonly used theories of failure for plane stress problems are presented:

For ductile materials:

- Maximum shear-stress theory
- Maximum distortion-energy theory

For brittle materials:

- Maximum normal-stress theory

Each of the criteria is presented in terms of principal stresses.

Maximum Shear-Stress Theory This theory states that a ductile material will yield when the absolute maximum shear stress in the material exceeds the yield shear stress τ_{cr} of the same

material under axial load; namely,

$$|\tau_{\max}| \leq \tau_{cr} = \frac{1}{2}\sigma_Y \tag{5.21}$$

where σ_Y is the yield stress of the material in a simple tension test. The fact $\tau_{cr} = \sigma_Y/2$ can be easily observed from Mohr's circle. For a biaxial (plane) stress problem, the criterion (5.21) is reduced to the following conditions on the in-plane principal stresses σ_1, σ_2:

(i) If σ_1 and σ_2 have the same sign

$$|\sigma_1| \leq \sigma_Y \quad \text{and} \quad |\sigma_2| \leq \sigma_Y \tag{5.22a}$$

(ii) If σ_1 and σ_2 have opposite signs

$$|\sigma_1 - \sigma_2| \leq \sigma_Y \tag{5.22b}$$

Maximum Distortion-Energy Theory This theory states that a ductile material will yield when the distortion energy per unit volume u_d of the material exceeds the distortion energy per unit volume $(u_d)_Y$ of the same material at the yield point in a simple tension test. For a plane stress problem,

$$u_d = \frac{1+\mu}{3E}\left(\sigma_1^2 - \sigma_1\sigma_2 + \sigma_2^2\right) \tag{5.23}$$

Also,

$$(u_d)_Y = \frac{1+\mu}{3E}\sigma_Y^2 \tag{5.24}$$

where σ_Y is the yield stress of the material in a simple tension test. It follows that the criterion for plane stress is

$$\sigma_1^2 - \sigma_1\sigma_2 + \sigma_2^2 \leq \sigma_Y^2 \tag{5.25}$$

Figures 5.2.12(a) to (c) show a graphic presentation of the maximum shear-stress theory and maximum energy-distortion theory, and their comparison. From Fig. 5.2.12(a) or Fig. 5.2.12(b), a ductile material will fail if the point (σ_1, σ_2) falls outside the shaded area.

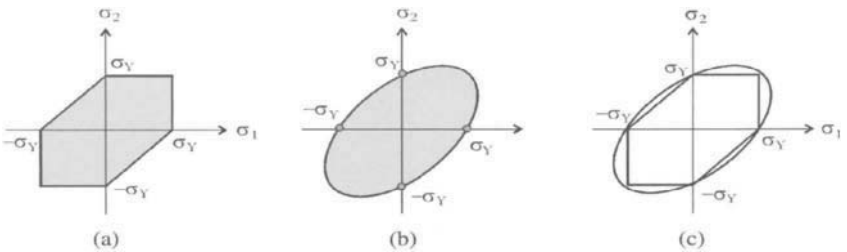


FIGURE 5.2.12 A graph of failure criteria: (a) maximum shear-stress theory; (b) maximum energy-distortion theory; (c) comparison of the theories.

Maximum Normal-Stress Theory The theory states that a brittle material will fail by fracture when the maximum principal stress in the material exceeds the ultimate normal stress σ_{ult} of the material in a simple tension test. For a plane stress problem, this can be translated into the following criteria

$$|\sigma_1| \leq \sigma_{ult} \quad \text{and} \quad |\sigma_2| \leq \sigma_{ult} \tag{5.26}$$

The Toolbox of this chapter provides a function `failure2D` for detecting the failure of a given material according to the above-mentioned criteria.

Window 2.7. Function `failure2D`

MATLAB Function: **`failure2D`**

Purpose:
To determine if a material under a given state of plane stress will fail.

Synopsis:

```
failure2D(stresses, S_threshold, Id_Criterion)
failure2D(stresses)
```

Description:
`failure2D(stresses, S_threshold, Id_Criterion)` tells if a material under a given state of stress will fail. Here, `stresses` is a vector of stress components, `stresses` = $[\sigma_x \ \sigma_y \ \tau_{xy}]$; `S_threshold` is the yield stress for a ductile material or the ultimate stress for a brittle material; and `Id_Criterion` is an integer identifying one of the following criteria to be used:

`Id_Criterion = 1`	maximum shear stress theory (for ductile materials)
`Id_Criterion = 2`	maximum distortion energy theory (for ductile materials)
`Id_Criterion = 3`	maximum normal stress theory (for brittle materials)

Function `failure2D` displays the results of failure analysis in the MATLAB command window.

`failure2D(stresses)` may be called without specifying which criterion to use. In this case, a question will pop up asking the user to identify a criterion and to assign a yield stress or ultimate stress.

EXAMPLE 2.7

In Fig. 5.2.13, a shaft of diameter $d = 0.024$ m is under an axial load $N = 80{,}000$ Newton and a torque $T = 620$ N-m. The shaft is made of aluminum material with a yield stress $\sigma_Y = 414$ MPa. Determine if the loadings will cause the shaft to fail by the maximum shear stress theory and the maximum distortion energy theory.

Solution. The maximum shear stress occurs at a point on the outer surface of the shaft. At point A, the stress components are expressed as

$$\sigma_x = \frac{N}{A} = \frac{N}{\pi d^2/4}, \quad \sigma_y = 0, \quad \tau_{xy} = -\frac{T \cdot d/2}{J} = -\frac{T \cdot d/2}{\pi d^4/32} = -\frac{16T}{\pi d^3}$$

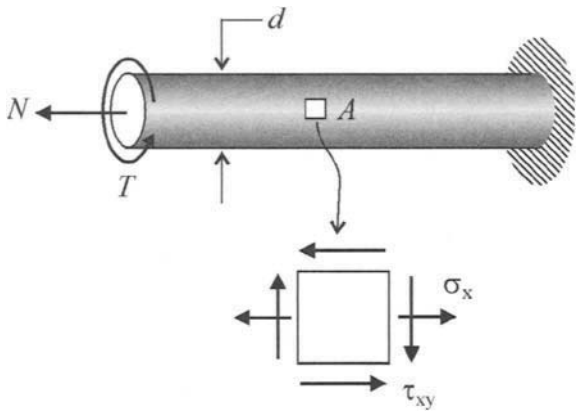


FIGURE 5.2.13

```
>> N = 80,000; T = 620; d = 0.024;
>> Sx = N/(pi*d^2/4); Sy = 0; Txy = -16*T/(pi*d^3);
>> stresses = [Sx Sy Txy];
>> S_yield = 414e6;
>> failure2D(stresses, S_yield, 1);
```

produces the following results:

```
Failure Analysis of Material in Plane Stress
--------------------------------------------
Maximum shear stress theory for ductile materials is used:
   |S1| <= S_yield and |S2| <= S_yield if S1 and S2 have the same
sign
   |S1 - S2| <= S_yield if S1 and S2 have opposite signs

Given stress components:
   normal stress Sx = 176838.8257
   normal stress Sy = 0
   shear stress Txy = -228416816.4745

Principal stresses:
   S1 = 228505253.0007 with ang_p1 = -44.9889 deg
   S2 = -228328414.1751 with ang_p2 = 45.0111 deg

Maximum in-plane shear stress:
   Tmax = 228416833.5879 with ang_T = 90.0111 deg and 0.01109 deg
   On the planes of Tmax, average normal stress Savg = (Sx + Sy)/2
= 88419.4128

Result and Conclusion
---------------------
Given yield stress S_yield = 414000000
S1*S2 < 0
|S1 - S2| = 456833667.1758 > S_yield
The criterion is not met and material failure will occur.
```

So, the shaft will fail by the maximum shear stress theory. On the other hand,

```
>> failure2D(stresses, S_yield, 2);
```

gives

```
Failure Analysis of Material in Plane Stress
----------------------------------------
Maximum distortion energy theory for ductile materials is used:
   S1^2 -S1*S2 + S2^2 <= S_yield^2

   Given yield stress S_yield = 414000000
   ud = S1^2 -S1*S2 + S2^2 = 1.565228e+017
   ud <= S_yield^2 = 1.713960e+017
   The criterion is met and material failure will not occur.
```

Thus, the shaft will not fail according to the maximum distortion energy theory.

5.3 Quick Solution Guide

MATLAB Functions

This chapter contains a toolbox of MATLAB functions for stress analysis in two-dimensional problems; see the table below. Refer to the corresponding window or section for the utility of each function.

	Stress	
`stressTF`	Transform stress components from an element to another of different orientation	Window 2.1
`pristress`	Compute principal stresses and maximum in-plane shear stress, and plot Mohr's circle for plane stress	Window 2.2
	Strain	
`strainTF`	Transform strain components from an element to another of different orientation	Window 2.3
`pristrain`	Compute principal strains and maximum in-plane shear strain, and plot Mohr's circle for plane strain	Window 2.4
`strgauges`	Determine strain components based on the readings from a set of three strain gauges	Window 2.5
	Material Properties	
`hooks2e`	Compute strain components based on Hooke's law	Window 2.6
`hooke2s`	Compute stress components based on Hooke's law	Window 2.6
`failure2D`	Tell if a material under a given state of in-plane stress will fail, based on one of three commonly used criteria of failure	Window 2.7
	Utilities	
`TBdemo`	Show how the Toolbox works and what it can do	Section 5.1
`TBinfo`	Show the information of the Toolbox	Section 5.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 5.1

5.4 References

1. Bedford A, Liechti K M 2000 *Mechanics of Materials*, Prentice-Hall: Upper Saddle River, New Jersey.
2. Gere J M, Timoshenko S P 1997 *Mechanics of Materials*, 4th ed., PWS Publishing Co.: Boston.
3. Popov E P 1998 *Engineering Mechanics of Solids*, 2nd ed., Prentice-Hall: Upper Saddle River, New Jersey.
4. Riley W F, Sturges L D, Morris D H 1999 *Mechanics of Materials*, 5th ed., John Wiley & Sons, Inc.: New York.
5. Shames I H, Pitarresi J M 2000 *Introduction to Solid Mechanics*, 3rd ed., Prentice-Hall: Upper Saddle River, New Jersey.

PART II
Structural Mechanics

This Page Intentionally Left Blank

6 Static Analysis of Constrained Multispan Beams

Inside
- Getting Started
- System Description
- Solution by the Toolbox
- The Distributed Transfer Function Method
- Quick Solution Guide
- References

6.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for static analysis of constrained, multispan Euler-Bernoulli beams.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **`TBdemo`** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

Quick Tutorial

To show how to use the Toolbox, consider a three-stepped beam in Fig. 6.1.1, which is subject to a pointwise force and a uniform load at its interior points, and a moment at its right end. To obtain the static response of the structure, in the MATLAB command window, type:

```
>> Beam_Spec = [80 2; 50 2; 30 1];
>> BC_Spec = [2 1];
>> setbeam2(Beam_Spec, BC_Spec)
>> Load_Spec = [1 1 1 2 -2 0 0; 3 0 0 -4 0 0 0];
>> BD_Spec = [0 0 0 -3.5];
>> beam2fbs(Load_Spec, BD_Spec)
```

where >> is the prompt in the MATLAB command window. This yields the spatial distribution of the displacement, rotation, bending moment, and shear force of the beam structure in Fig. 6.1.2. Also, the maximum response of the structure and the reactions at the supports are displayed in the MATLAB command window as follows:

```
Maximum Response in Absolute Value
----------------------------------
   Max displacement = -0.022196, location x = 2.5
   Max slope (degrees) = -1.6455, location x = 5
   Max bending moment = -3.5, location x = 5
   Max shear force = -4.0913, location x = 4

Reactions at the Nodes of the Beam Structure
--------------------------------------------
   At left boundary:
      Mc = 2.0437, Rc = 1.9087
      w0 = 0, angle0 = 0 degrees

   At node 1:
      Mc = 0, Rc = 0
      w = -0.020322, angle = -0.43159 degrees

   At node 2:
      Mc = 0, Rc = 0
      w = -0.0068853, angle = 1.1312 degrees
```

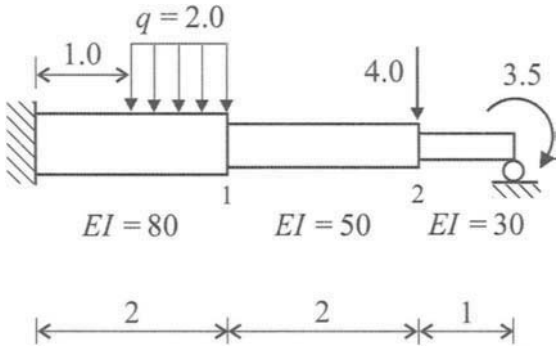


FIGURE 6.1.1 A three-stepped beam.

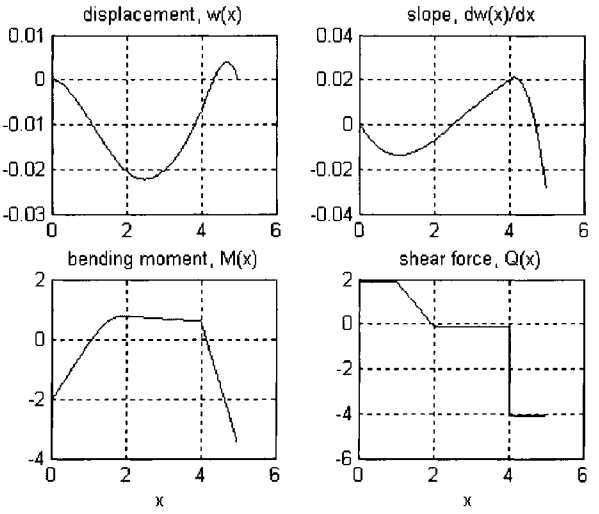


FIGURE 6.1.2 Beam response.

```
At right boundary:
   Mc = -3.5, Rc = 4.0913
   wL = 0, angleL = -1.6286 degrees
```

In the above static analysis, the following two functions from the Toolbox are called:

(a) Function **setbeam2** that inputs the beam parameters and sets up the boundary conditions and constraints; and
(b) Function **beam2fbs** that computes the response of the beam structure under the external and boundary disturbances and displays the computed results.

These functions, along with others from the Toolbox, will be described in the subsequent sections.

For a quick solution, the reader may refer directly to the Quick Solution Guide (Section 6.5), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To understand better how the Toolbox works, the reader is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 6.6.

6.2 System Description

The beam structure in consideration is an assembly of N uniform Euler-Bernoulli beam elements or segments that are serially connected at N-1 points called *nodes*; see Fig. 6.2.1. At those nodes, there may be constrains like hinges and springs. The two ends of the beam structure are Nodes 0 and N.

Beam Elements

The transverse displacement (deflection) $w(x)$ of the ith beam element is governed by the differential equation of Euler-Bernoulli beam model (Reference 1)

$$EI_i \frac{d^4}{dx^4} w(x) = f(x), \quad 0 \le x \le l_i \tag{6.1}$$

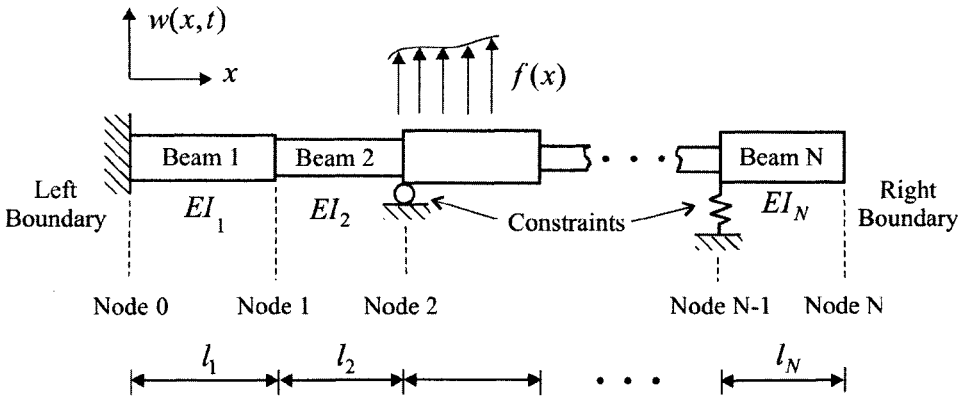


FIGURE 6.2.1 Schematic of a constrained multispan beam.

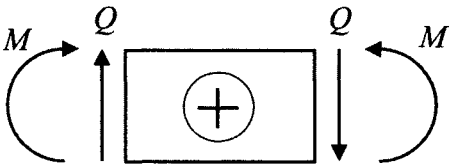


FIGURE 6.2.2 Sign convention of bending moment and shear force.

where EI_i and l_i are the bending stiffness and length of the beam element, respectively, $f(x)$ is an external force, and x is a local coordinate of the beam element. The sign convention for beam displacement and external forces is as follows. The positive direction of beam displacement and transverse forces (either pointwise or distributed) is upward (↑); the positive direction of a torque is counterclockwise (↺). The internal forces of the beam element are

$$\text{bending moment:} \qquad M(x) = EI_i \frac{d^2w(x)}{dx^2} \tag{6.2}$$

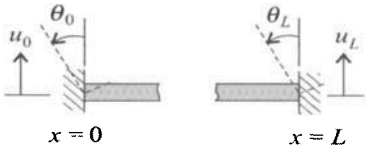
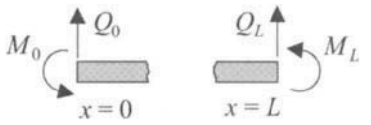
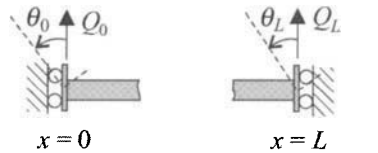
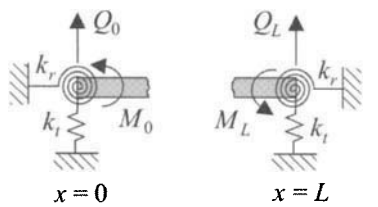
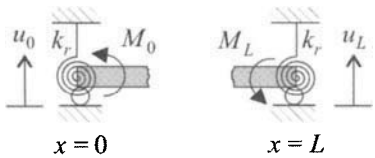
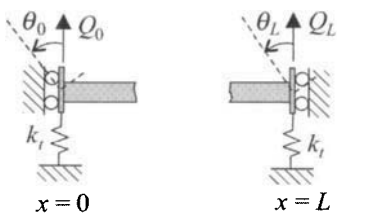
$$\text{shear force:} \qquad Q(x) = EI_i \frac{d^3w(x)}{dx^3} \tag{6.3}$$

Figure 6.2.2 shows the sign convention of bending moment and shear force. In addition, the rotation angle of the beam cross-section at x is $\theta(x) = \frac{dw(x)}{dx}$.

Boundary Conditions and Boundary Disturbances

Seven types of boundary conditions with disturbances are given in Table 6.2.1, where u_0 and u_L are prescribed transverse displacements at the boundaries, θ_0 and θ_L rotations, M_0 and M_L moments, and Q_0 and Q_L transverse forces. Here L is the total length of the beam structure, i.e., $L = l_1 + l_2 + \cdots + l_N$. The positive direction of displacements and transverse forces is upward; the positive direction of rotations and moments is counterclockwise.

TABLE 6.2.1 Boundary Conditions and Boundary Disturbances

Boundary Type	*Boundary Conditions*
(B1) Pinned or hinged end	Left end: $w(0)=u_0,\quad -EI\dfrac{d^2w(0)}{dx^2}=M_0$ Right end: $w(L)=u_L,\quad EI\dfrac{d^2w(L)}{dx^2}=M_L$
(B2) Clamped or fixed end 	Left end: $w(0)=u_0,\quad \dfrac{dw(0)}{dx}=\theta_0$ Right end: $w(L)=u_L,\quad \dfrac{dw(L)}{dx}=\theta_L$
(B3) Free end 	Left end: $-EI\dfrac{d^2w(0)}{dx^2}=M_0,\quad EI\dfrac{d^3w(0)}{dx^3}=Q_0$ Right end: $EI\dfrac{d^2w(L)}{dx^2}=M_L,\quad -EI\dfrac{d^3w(L)}{dx^3}=Q_L$
(B4) Sliding or guided end 	Left end: $\dfrac{dw(0)}{dx}=\theta_0,\quad EI\dfrac{d^3w(0)}{dx^3}=Q_0$ Right end: $\dfrac{dw(L)}{dx}=\theta_L,\quad -EI\dfrac{d^3w(L)}{dx^3}=Q_L$
(B5) Elastic end 	Left end: $k_r\dfrac{dw(0)}{dx}-EI\dfrac{d^2w(0)}{dx^2}=M_0,\quad k_tw(0)+EI\dfrac{d^3w(0)}{dx^3}=Q_0$ Right end: $k_r\dfrac{dw(L)}{dx}+EI\dfrac{d^2w(L)}{dx^2}=M_L,\quad k_tw(L)-EI\dfrac{d^3w(L)}{dx^3}=Q_L$
(B6) Hinged-elastic end 	Left end: $w(0)=u_0,\quad k_r\dfrac{dw(0)}{dx}-EI\dfrac{d^2w(0)}{dx^2}=M_0$ Right end: $w(L)=u_L,\quad k_r\dfrac{dw(L)}{dx}+EI\dfrac{d^2w(L)}{dx^2}=M_L$
(B7) Sliding-elastic end 	Left end: $\dfrac{dw(0)}{dx}=\theta_0,\quad k_tw(0)+EI\dfrac{d^3w(0)}{dx^3}=Q_0$ Right end: $\dfrac{dw(L)}{dx}=\theta_L,\quad k_tw(L)-EI\dfrac{d^3w(L)}{dx^3}=Q_L$

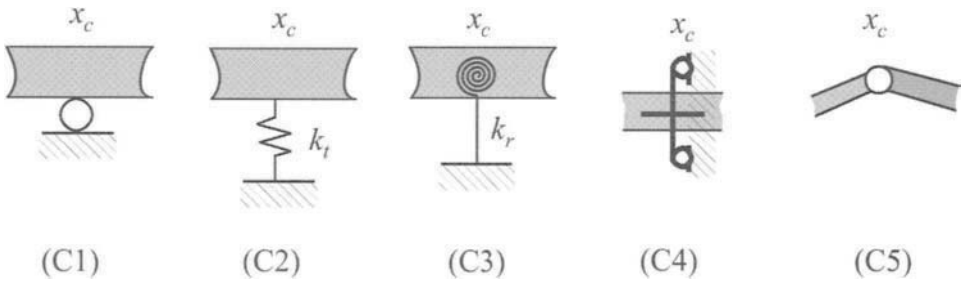


FIGURE 6.2.3 Five types of constraints.

Constraints

Constraints are specified at the nodes of a beam structure. Five types of constraints are considered here; see Fig. 6.2.3, where x_c represents a node.

(C1) *Hinge.* At x_c, the beam displacement is zero; i.e., $w(x_c) = 0$.

(C2) *Translational spring.* Resisting the beam displacement at x_c, the spring applies to the beam a transverse constraint force $R_c = -k_t w(x_c)$.

(C3) *Torsional spring.* Resisting the beam rotation at x_c, the spring applies to the beam a constraint moment $M_c = -k_r \theta(x_c)$, where $\theta(x_c) = \frac{d}{dx} w(x_c)$ is the rotation angle of the beam at x_c.

(C4) *Sliding constraint.* The beam is prevented from rotation at x_c; i.e., $\theta(x_c) = \frac{d}{dx} w(x_c) = 0$.

(C5) *Hinge connection.* Two adjacent beam elements are hinged at x_c such that they have the same transverse displacement at the node. However, these elements may have different rotation angles at x_c, namely, $\theta(x_c - 0) \neq \theta(x_c + 0)$. In this case, the hinge cannot generate any constraint moment.

At a node, two constraints, like C2 and C3 or C1 and C3, can be simultaneously imposed, as long as the node still has at least one degree of freedom. Note that constraints C1, C4, and C5 each remove one degree of freedom, while C2 and C3 don't.

Settlement of Supports

Constraints C1 to C4, which exert reaction forces and moments to a beam structure, are also called *supports*. As shown in Fig. 6.2.4, the foundation of these supports may experience transverse deflection u_c or rotation θ_c. The description of the displacements at the foundation of types C1 to C4 is as follows:

(C1) Hinge. At node x_c, the beam displacement is $w(x_c) = u_c$.

(C2) Translational spring. At node x_c, the spring applies to the beam a transverse constraint force $R_c = -k_t (w(x_c) - u_c)$.

(C3) Torsional spring. At node x_c, the spring applies to the beam a constraint moment $M_c = -k_r \left(\frac{d}{dx} w(x_c) - \theta_c \right)$.

(C4) Sliding constraint. At node x_c, the rotation of the beam is $\frac{d}{dx} w(x_c) = \theta_c$.

Displacements at the supports of a structure are often referred as *settlement of supports*. For instance, a support of a multispan bridge that has a downward deflection Δ can be described by constraint C1 with $u_c = -\Delta$. Under support displacements, a beam structure may deform,

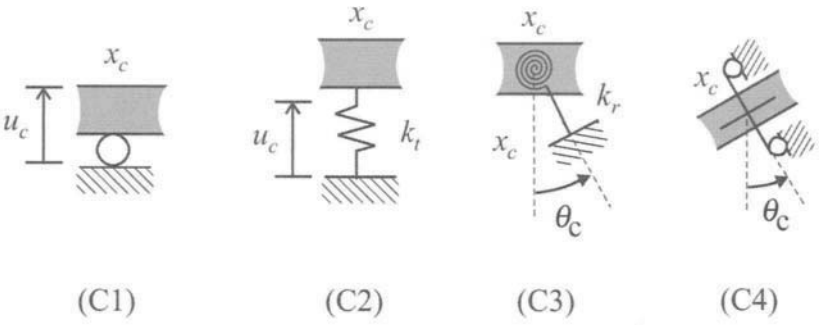


FIGURE 6.2.4 Support displacements at constraints C1 to C4.

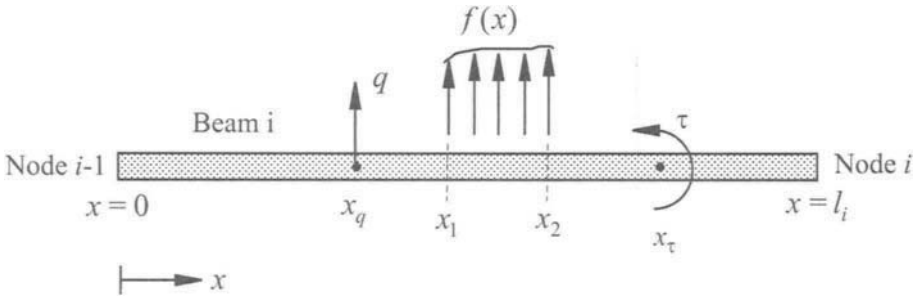


FIGURE 6.2.5 Three types of external loads.

resulting in nonzero bending moment and shear force. This is true for statically indeterminate structures; see Example 3.5 in Section 6.3.4.

External Loads

External loads can be described in the local coordinate system of each individual beam element. In Fig. 6.2.5, the ith beam element is subject to three types of external loads:

(a) Pointwise force q at point x_q: $f(x) = q\delta(x - x_q)$, where $\delta(\cdot)$ is the delta function. For a force applied at node i-1 (or node i), $x_q = 0$ (or $x_q = l_i$).
(b) Moment τ at point x_τ: $f(x) = -\tau \frac{d}{dx}\delta(x - x_\tau)$.
(c) Distributed force in the region $x_1 \le x \le x_2$, where $0 \le x_1 < x_2 \le l_1$.

Reactions at Supports

At an interior node (node number = 1, 2, . . . , N-1) of a multispan beam, if one of constraints C1 to C4 in Fig. 6.2.3 is imposed, the constraint provides a support to the beam. The support will exert a reaction force R_c and/or a moment M_c to the beam, depending on the type of constraint; see Fig. 6.2.6, for instance. These reaction forces are expressed as

$$R_c = \left(EI\frac{d^3}{dx^3}w\right)_{x=x_c+0} - \left(EI\frac{d^3}{dx^3}w\right)_{x=x_c-0} - q_e$$

$$M_c = -\left(EI\frac{d^2}{dx^2}w\right)_{x=x_c+0} + \left(EI\frac{d^2}{dx^2}w\right)_{x=x_c-0} - \tau_e$$

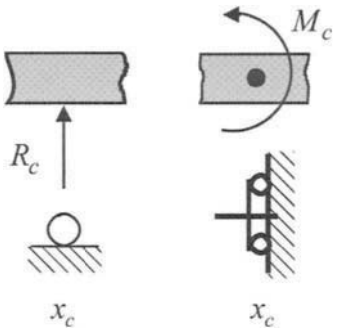


FIGURE 6.2.6 Reactions at supports.

where x_c represents the location of the node, and q_e and τ_e are a pointwise external load and an external torque applied to the beam structure at x_c (not shown in Fig. 6.2.6). At the two ends of the beam structure, the reactions are given by

$$\text{At } x = 0 \text{ (Node 0):} \quad R_c = EI\frac{d^3}{dx^3}w(0) - Q_0, \qquad M_c = -EI\frac{d^2}{dx^2}w(0) - M_0$$

$$\text{At } x = L \text{ (Node N):} \quad R_c = -EI\frac{d^3}{dx^3}w(L) - Q_L, \quad M_c = EI\frac{d^2}{dx^2}w(L) - M_L$$

where Q_0 and M_0 are the external force and moment applied at Node 0, and Q_L and M_L are the external force and moment applied at Node N, which are shown in Table 6.2.1.

Static Problem

A fundamental problem of a constrained multispan beam structure is to determine its static response (displacement $w(x)$, rotation $\theta(x)$, bending moment $M(x)$, and shear force $Q(x)$) subject to external loads, boundary conditions, and support displacements. This problem can be solved by the MATLAB functions from the Toolbox of this chapter; see Section 6.3.

6.3 Solution by the Toolbox

This chapter has a toolbox of MATLAB functions for static analysis of constrained multispan beams. These functions are developed based on an exact analytical solution method described in Section 6.4. Static analysis by the Toolbox takes the following two steps:

Step 1. Set up the beam structure by function **`setbeam2`**.
Step 2. Determine and plot the system response by function **`beam2fbs`**.

These functions, along with others from the Toolbox, are presented in sequel.

6.3.1 System Setup

Consider a constrained multispan beam of N elements as shown in Fig. 6.2.1, in which element numbers and node numbers are assigned from left to right in an increasing order. In static analysis of such a beam structure, function `setbeam2` is called first to specify beam parameters, boundary conditions, and constrains (see Window 3.1).

Window 3.1. Function setbeam2

MATLAB Function: **setbeam2**

Purpose:
To set up beam parameters, boundary conditions, and constraints for a multispan beam.

Synopsis:

```
setbeam2(Beam_Spec, BC_Spec, Constr_Spec)
setbeam2(Beam_Spec, BC_Spec, Constr_Spec, n)
```

Description:
`setbeam2(Beam_Spec, BC_Spec, Constr_Spec, n)` undertakes the following tasks:

(a) It inputs the bending stiffness and length of beam elements by a matrix `Beam_Spec`;
(b) It specifies the boundary conditions of the structure by a vector `BC_Spec`; and
(c) It imposes constraints onto the structure by a matrix `Constr_Spec`.

Upon completion of those tasks, `setbeam2` displays the information on the beam structure.

`setbeam2(Beam_Spec, BC_Spec, Constr_Spec, n)` undertakes the above-mentioned tasks and sets up `n` equally-spaced points along each beam element for computation and display purposes. By default, `n` = 101.

If the beam structure has no constrains, set `Constr_Spec = 0` or replace it by `[]` as a placeholder. For instance, `setbeam2(Beam_Spec, BC_Spec)` specifies a beam structure whose interior nodes (Nodes 1 to N = 1) are not constrained.

Function `setbeam2` assigns the parameters, boundary conditions, and constraints for an N-element beam structure by the following vector and matrices:

(i) The parameters of the beam elements are assigned by an $N \times 2$ matrix

$$\texttt{Beam\_Spec} = \begin{bmatrix} EI_1 & l_1 \\ EI_2 & l_2 \\ \vdots & \vdots \\ EI_N & l_N \end{bmatrix}$$

where the ith row contains the bending stiffness and length of the ith element.

(ii) The boundary conditions of the structure (at Nodes 0 and N) are assigned by a row vector

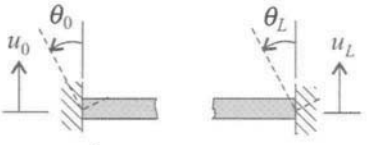
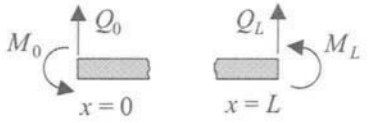
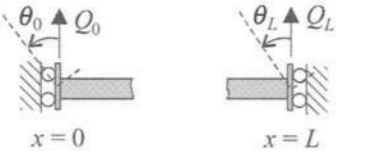
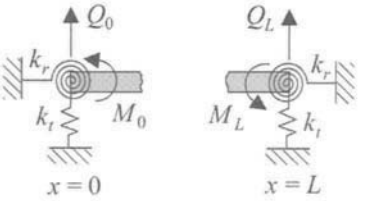
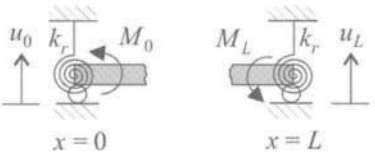
```
BC_Spec = [BC_Left   BC_Right]
```

where `BC_Left` and `BC_Right` are two row subvectors describing the boundary conditions at the left end (Node 0) and the right end (Node N) of the beam structure, respectively. Table 6.3.1 lists the entries of these subvectors for the seven different

TABLE 6.3.1 Boundary Conditions and Boundary Disturbances

Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`

Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

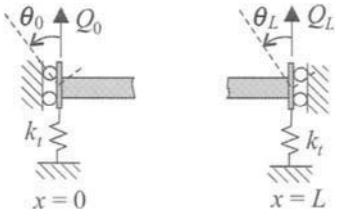
Boundary Condition	*Left Boundary ($x = 0$)*		*Right Boundary ($x = L$)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B1) Pinned or hinged end	[1]	$[u_0 \quad M_0]$	[1]	$[u_L \quad M_L]$
(B2) Clamped or fixed end 	[2]	$[u_0 \quad \theta_0]$	[2]	$[u_L \quad \theta_L]$
(B3) Free end 	[3]	$[M_0 \quad Q_0]$	[3]	$[M_L \quad Q_L]$
(B4) Sliding or guided end 	[4]	$[\theta_0 \quad Q_0]$	[4]	$[\theta_L \quad Q_L]$
(B5) Elastic end 	[5 k_r k_t]	$[M_0 \quad Q_0]$	[5 k_r k_t]	$[M_L \quad Q_L]$
(B6) Hinged-elastic end 	[6 k_r]	$[u_0 \quad M_0]$	[6 k_r]	$[u_L \quad M_L]$

(continued)

TABLE 6.3.1 Boundary Conditions and Boundary Disturbances (*Continued*)

Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`

Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

Boundary Condition	*Left Boundary ($x = 0$)*		*Right Boundary ($x = L$)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B7) Sliding-elastic end 	$[7 \quad k_t]$	$[\theta_0 \quad Q_0]$	$[7 \quad k_t]$	$[\theta_L \quad Q_L]$

u_0, u_L—end displacement; θ_0, θ_L—end rotation in radians;
M_0, M_L—end moment; Q_0, Q_L—end transverse force;
k_r—torsional spring coefficient; k_t—translational spring coefficient.
Positive direction of end displacement and transverse force: upward
Positive direction of end rotation and moment: counterclockwise

boundary conditions (B1 to B7). As can be seen, the first element of `BC_Left` and `BC_Right` is an integer from 1 to 7 identifying the boundary type.

As an example, the boundary conditions of a pinned-clamped beam structure is described by the row vector

```
BC_Spec = [1 2]
```

whose subvectors are `BC_Left = [1]` and `BC_Right = [2]`.

(iii) Constraints are imposed at the interior nodes of the beam structure by a matrix

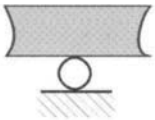
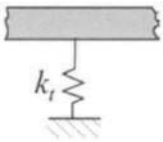
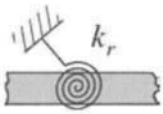
$$\texttt{Constr\_Spec} = \begin{bmatrix} \texttt{Constr\_1} \\ \texttt{Constr\_2} \\ \vdots \\ \texttt{Constr\_Nc} \end{bmatrix}$$

where Nc is the number of constraints. The jth row vector of the matrix specifies the jth constraint, and is of the form

```
Constr_j = [node_no   constr_type_no   p]
```

where `node_no` is node number (1, 2, . . . , N-1), `constr_type_no` is an integer between 1 and 5 specifying one of the five types of constraints (C1 to C5) shown in Fig. 6.2.3, and the parameter `p` is a spring coefficient for constraints C2 and C3, and zero for others. Table 6.3.2 lists the entries of `Constr_j` for these five types of constraints. If the beam structure has no constraints at any of its interior nodes 1, 2, . . . , N-1, simply set `Constr_Spec = 0` or `Constr_Spec = []`.

TABLE 6.3.2 Specification of Constraints

Constraint Type		Constr_j
(C1) Hinge		[node_no 1 0]
(C2) Translational spring		[node_no 2 k_t]
(C3) Torsional spring		[node_no 3 k_r]
(C4) Sliding constraint		[node_no 4 0]
(C5) Hinge connection		[node_no 5 0]

Once function `setbeam2` is used, the information of the beam parameters and configuration can be accessed at any time by calling function `systinfo`, namely,

```
>> systinfo
```

EXAMPLE 3.1

For the four-element beam structure in Fig. 6.3.1, the input arguments of function `setbeam2` are

$$\texttt{Beam\_Spec} = \begin{bmatrix} 60 & 1.5 \\ 60 & 1.0 \\ 25 & 1.0 \\ 25 & 0.7 \end{bmatrix}, \quad \texttt{BC\_Spec} = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad \texttt{Constr\_Spec} = \begin{bmatrix} 1 & 2 & 50 \\ 3 & 4 & 0 \end{bmatrix}.$$

The beam structure is set up by

```
>> Beam_Spec = [60 1.5; 60 1; 25 1; 25 0.7];
>> BC_Spec = [1 1];
>> Constr_Spec =[1 2 50; 3 4 0]
>> setbeam2(Beam_Spec, BC_Spec, Constr_Spec)
```

which shows the following information

```
STATIC ANALYSIS OF A CONSTRAINED MULTISPAN BEAM STRUCTURE

Description of the Beam Structure
---------------------------------
Number of beam elements = 4
   Beam 1: EI = 60, L = 1.5
   Beam 2: EI = 60, L = 1
   Beam 3: EI = 25, L = 1
   Beam 4: EI = 25, L = 0.7

Total length of the beam structure = 4.2

Boundary conditions:
   Left end: pinned
   Right end: pinned

Number of constraints = 2
   At node 1, C2: translational spring, kt = 50
   At node 3, C4: sliding, dw/dx = 0

Number of spatial points of simulation for each beam element:
n = 101
```

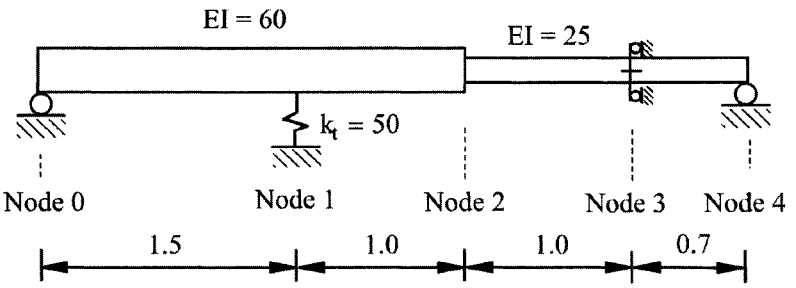


FIGURE 6.3.1

6.3.2 Response to External Forces

The Toolbox of this chapter provides a function `beam2fbs` for computing the static response (displacement, rotation, bending moment, and shear force) of a multispan beam structure under external forces; see Window 3.2.

Window 3.2. Function `beam2fbs` for Response to External Loads

MATLAB Function: **`beam2fbs`**

Purpose:
To compute and display the static response of a multispan beam under external forces.

Window 3.2. Function beam2fbs for Response to External Loads (*Continued*)

Synopsis:

```
beam2fbs(Load_Spec)
[y, r] = beam2fbs(Load_Spec)
```

Description:

`beam2fbs(Load_Spec)` plots the response of the beam structure subject to `Nf` external loads that are specified by an `Nf`×7matrix

$$\texttt{Load\_Spec} = \begin{bmatrix} \texttt{Load\_1} \\ \texttt{Load\_2} \\ \vdots \\ \texttt{Load\_Nf} \end{bmatrix}$$

where each row of the matrix specifies an external load. A typical row is of the form

```
Load_j = [beam_no load_type p1 p2 p3 p4 p5]
```

where `beam_no` is a beam element number (`beam_no` = 1 or 2 ... or N) indicating which beam the load is applied to, `load_type` is an integer between −1 to 3 identifying one of the five types of external loads given in Table 6.3.3, and `p1` to `p5` are parameters describing the location and form of the load. Function `beam2fbs` also displays the maximum response and reactions at supports.

`[y, r] = beam2fbs(Load_Spec)` returns the computed beam response in a three-dimensional matrix `y`, and the reactions and displacements at the nodes of the beam structure in a matrix `r`. With `y`, the beam response can be plotted by function `plotbeam2(y)`; see Window 3.6.

Table 6.3.3 lists five types of external loads considered in the Toolbox. From the table, the second element of the vector `Load_j` is assigned as follows:

`load_type = 0` for a pointwise force (L1);
`load_type = -1` for a torque (L2);
`load_type = 1 to 3` for spatially distributed loads (D1 to D3).

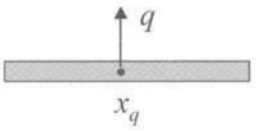
The x_q, x_τ, x_1, and x_2 are the local coordinate parameters measured from the left end of a particular beam element. These parameters describe the location or region of distribution of the load. In addition, $\delta(\cdot)$ in L1 and L2 is the delta function.

EXAMPLE 3.2

The stepped beam in Fig. 6.3.2 is subject to three external loads: a pointwise load ($q = -10$) applied to beam element 1 at $x_q = 0.72$; a torque ($\tau = 6$) applied to beam element 2 at $x_\tau = 0.5$; and a linearly distributed load applied to beam element 3

TABLE 6.3.3 Specification of External Loads

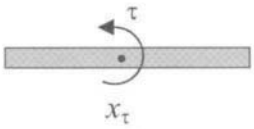
(L1) Pointwise force



$f(x) = q\delta(x - x_q)$

`Load_j = [beam_no 0 xq q 0 0 0]`

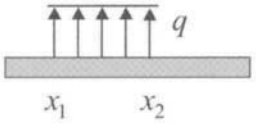
(L2) Torque



$f(x) = -\tau \dfrac{d}{dx}\delta(x - x_\tau)$

`Load_j = [beam_no -1 xτ τ 0 0 0]`

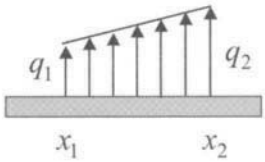
(D1) Uniform distribution



$f(x) = q \quad \text{for } x_1 \le x \le x_2$

`Load_j = [beam_no 1 x1 x2 q 0 0 0]`

(D2) Linear distribution



$f(x) = q_1 + \dfrac{q_2 - q_1}{a}(x - x_1) \quad \text{for } x_1 \le x \le x_2$

where $a = x_2 - x_1$

`Load_j = [beam_no 2 x1 x2 q1 q2 0]`

(D3) Quadratic distribution

$f(x) = c_0 + c_1 x + c_2 x^2 \quad \text{for } x_1 \le x \le x_2$

`Load_j = [beam_no 3 x1 x2 c0 c1 c2]`

Sign Convention

The positive direction of transverse loads (L1, D1 to D3) is upward.

The positive direction of a torque (L2) is counterclockwise.

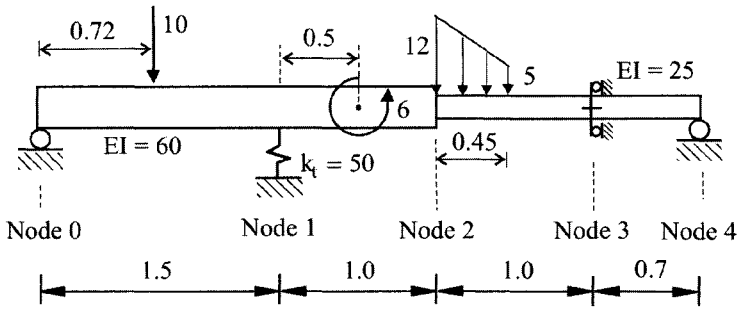


FIGURE 6.3.2

with $q_1 = -12$, $q_2 = -5$, $x_1 = 0$, and $x_2 = 0.45$. The load specification matrix for function `beam2fbs` is

$$\texttt{Load\_Spec} = \begin{bmatrix} 1 & 0 & 0.72 & -10 & 0 & 0 & 0 \\ 2 & -1 & 0.5 & 6 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0.45 & -12 & -5 & 0 \end{bmatrix}$$

By

```
>> beam = [60 1.5; 60 1; 25 1; 25 0.7]; bc = [1 1];
>> constr = [1 2 50; 3 4 0];
>> setbeam2(beam, bc, constr)
>> Load_Spec = [1 0 0.72 -10 0 0 0; 2 -1 0.5 6 0 0 0;
   3 2 0 0.45 -12 -5 0];
>> beam2fbs(Load_Spec)
```

the static response of the beam structure is plotted in Fig. 6.3.3. Function beam2fbs also displays the maximum response and support reactions as follows:

```
Maximum Response in Absolute Value
----------------------------------
Max displacement = -0.07207, location x = 1.365
Max slope (degrees) = -5.0578, location x = 0
Max bending moment = 5.6949, location x = 0.72
Max shear force = 7.9096, location x = 0

Reactions at the Nodes of the Beam Structure:
---------------------------------------------
At left boundary:   Mc = 0, Rc = 7.9096
                  w0 = 0, angle0 = -5.0578 degrees
At node 1:          Mc = 0, Rc = 3.5721
                  w = -0.071442, angle = 0.5346 degrees
At node 2:          Mc = 0, Rc = 0
                  w = -0.036624, angle = 2.2574 degrees
At node 3:          Mc = -3.6949, Rc = 0
                  w = -0.010717, angle = 0 degrees
At right boundary:  Mc = 0, Rc = 2.3433
                  wL = 0, angleL = 1.3155 degrees
```

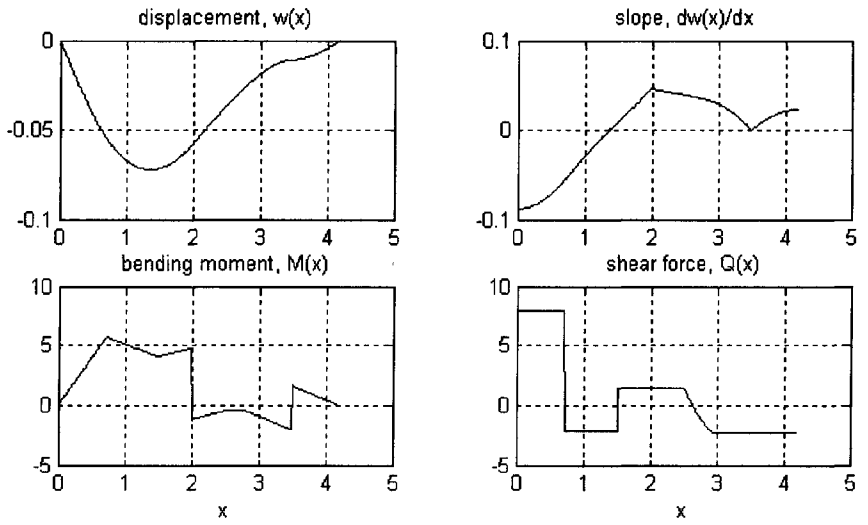


FIGURE 6.3.3

EXAMPLE 3.3

In Fig. 6.3.4, a beam structure is composed of three flexible beam elements and one rigid bar. The first and second beam elements are hinge-connected (constraint C5). There are two external loads: a linearly distributed load applied to the first beam element and a pointwise force applied to the fourth element. The matrices specifying the beam structure are

$$\texttt{Beam\_Spec} = \begin{bmatrix} 20 & 2 \\ 10 & 1 \\ 10^8 & 1 \\ 15 & 2 \end{bmatrix}, \quad \texttt{BC\_Spec} = [2 \quad 2], \quad \texttt{Constr\_Spec} = \begin{bmatrix} 1 & 5 & 0 \\ 3 & 1 & 0 \end{bmatrix}$$

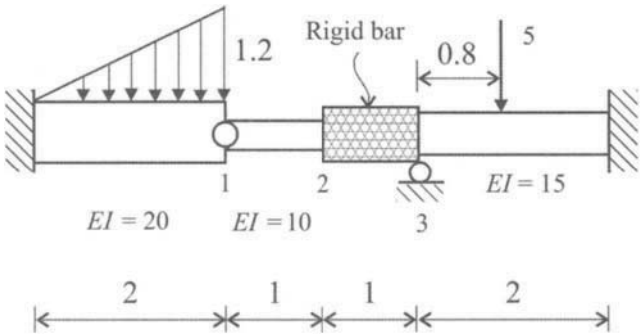


FIGURE 6.3.4

where a flexible beam element of a very large bending stiffness (10^8) is used to model the rigid bar. By the MATLAB commands

```
>> beam = [20 2; 10 1; 1e8 1; 15 2]; bc = [2 2];
>> constr = [1 5 0; 3 1 0];
>> setbeam2(beam, bc, constr);
>> Load_Spec = [1 2 0 2 0 -1.2 0; 4 0 0.8 -5 0 0 0];
>> beam2fbs(Load_Spec)
```

the static response of the beam structure is plotted in Fig. 6.3.5, and the reaction forces and displacements are displayed as follows:

```
Reactions at the Nodes of the Beam Structure
--------------------------------------------
At left boundary:   Mc = 0.37333,  Rc = 0.58667
              w0 = 0,  angle0 = 0 degrees

At node 1:          Mc = 0,  Rc = 0
              hinge connection (C5): w = -0.0062222
              angle- = 0.076394 degrees,  angle+ = 1.3494 degrees

At node 2:          Mc = 1.1865e-010,  Rc = -8.1254e-010
              w = 0.0071111,  angle = -0.40743 degrees
```

```
At node 3:          Mc = -3.2011e-011,  Rc = 3.6933
                w = 0,  angle = -0.40743 degrees
At right boundary:  Mc = -1.0667,  Rc = 1.92
                wL = 0,  angleL = 0 degrees
```

Note that at node 1 (hinge connection), the rotation angles of the adjacent elements are different.

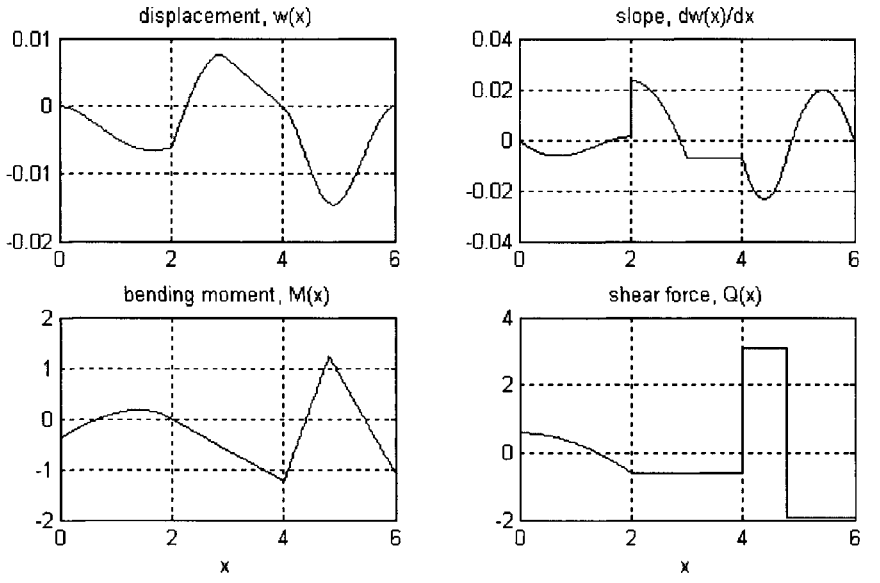


FIGURE 6.3.5

6.3.3 Response to Boundary Disturbances

To compute the response of a multispan beam structure subject to prescribed disturbances (displacements and forces) at its two ends, use function `beam2fbs` according to Window 3.3.

Window 3.3. Function beam2fbs for Response to Boundary Disturbances

MATLAB Function: **beam2fbs**

Purpose:
To compute the static response of a multispan beam structure under prescribed displacements and forces at the boundaries.

Synopsis:

```
beam2fbs(0, BD_Spec)
[y, r] = beam2fbs(0, BD_Spec)
```

Window 3.3. Function beam2fbs for Response to Boundary Disturbances (*Continued*)

Description:
`beam2fbs(0, BD_Spec)` plots the response of the beam structure subject to boundary disturbances that are specified by a row vector

```
BD_Spec = [BD_Left BD_Right]
```

where `BD_Left` and `BD_Right` are subvectors describing the disturbances at the left and right ends of the structure according to Table 6.3.1. Note that the boundary rotations θ_0 and θ_L are given in radians. Function `beam2fbs` also displays the maximum response and reactions at supports.

`[y, r] = beam2fbs(0, BD_Spec)` returns the computed beam response in a three-dimensional matrix `y` and the reactions and displacements at the nodes of the beam structure in a matrix `r`. With `y`, the beam response can be plotted by function `plotbeam2(y)`; see Window 3.6.

EXAMPLE 3.4

In Fig. 6.3.6, a two-span beam structure is subject to a prescribed displacement $u_0 = 0.02$ at its left end, and a moment $M_L = 4.7$ at its right end. The boundary disturbance vector by Table 6.3.1 is

```
BD_Spec = [0.02 0 0 4.7].
```

Typing the following in the MATLAB command window

```
>> beam_spec = [100 5; 45 6]; bc = [2 1]; constr = [1 1 0];
>> setbeam2(beam_spec, bc, constr)
>> bd =[0.02 0 0 4.6];
>> beam2fbs(0, bd)
```

yields the response of the beam structure plotted in Fig. 6.3.7.

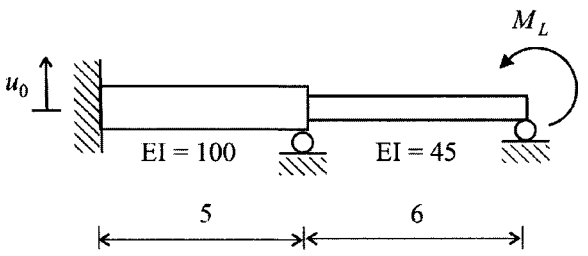


FIGURE 6.3.6

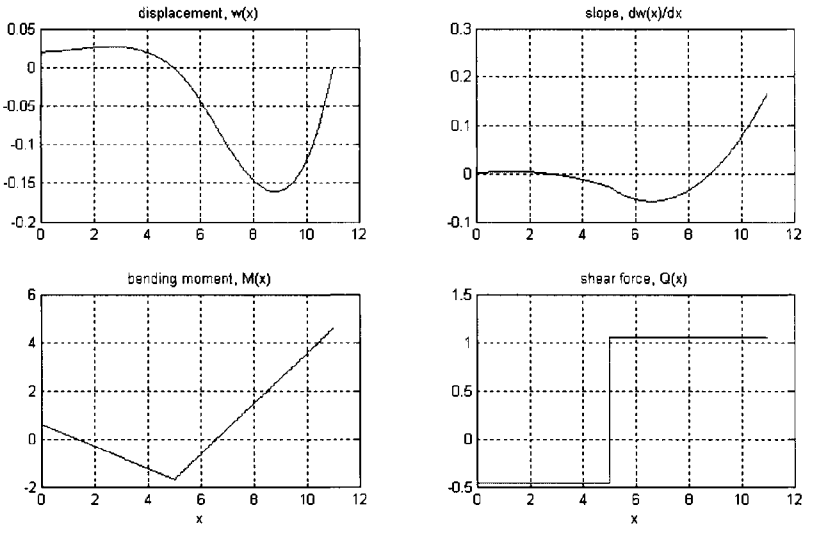


FIGURE 6.3.7

6.3.4 Response to Support Settlement

As discussed in Section 6.2, the supports of a beam structure can experience displacements, which can cause the deformation of the structure. Figure 6.3.8 shows an example, in which the beam structure is under a deflection u_c at the hinge support and a rotation θ_c at the sliding support. Function `beam2fbs` from this Toolbox can be used to compute the static response of multispan beam structures with support settlement; see Window 3.4.

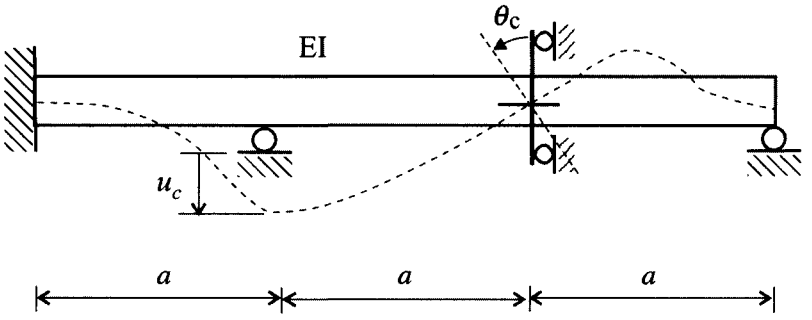


FIGURE 6.3.8 A beam structure with support settlement.

Window 3.4. Function `beam2fbs` for Support Settlement

MATLAB Function: **`beam2fbs`**

Purpose:
To compute the static response of a multispan beam structure under prescribed support settlement.

Window 3.4. Function beam2fbs for support settlement (*Continued*)

Synopsis:

```
beam2fbs(0, 0, SD_Spec)
[y, r] = beam2fbs(0, 0, SD_Spec)
```

Description:

`beam2fbs(0, 0, SD_Spec)` plots the response of the beam structure subject to settlement of supports. The support settlement is described by an `Ns`×3 matrix

$$\texttt{SD\_Spec} = \begin{bmatrix} \texttt{SD\_1} \\ \texttt{SD\_2} \\ \vdots \\ \texttt{SD\_Ns} \end{bmatrix}$$

where `Ns` is the number of support displacements and each row of the matrix specifies a support displacement. A typical row of `SD_Spec` has the form

```
SD_j = [node_no constr_type_no p]
```

where `node_no` is node number, `constr_type_no` is an integer between 1 and 4 specifying one of the four types of constraints (C1 to C4), and `p` is the specified support displacement or rotation. See Table 6.3.4 for the specification of `SD_j`. Function `beam2fbs` also displays the reactions at supports and maximum response.

`[y, r] = beam2fbs(0, 0, SD_Spec)` returns the computed beam response in a three-dimensional matrix `y` and the reactions and displacements at the nodes of the beam structure in a matrix `r`. With `y`, the beam response can be plotted by function `plotbeam2(y)`; see Window 3.6.

EXAMPLE 3.5

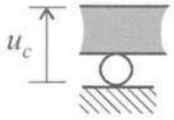
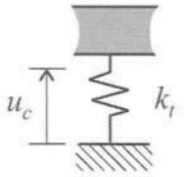
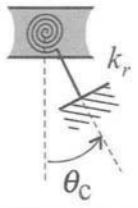
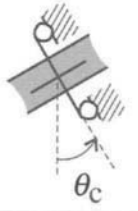
Consider the beam structure in Fig. 6.3.8, with EI = 40 and a = 5. Let the support displacements be $u_c = -0.08$ and $\theta_c = 0.05$. The displacements at the supports are assigned by the matrix

$$\texttt{SD\_Spec} = \begin{bmatrix} 1 & 1 & -0.08 \\ 2 & 4 & 0.05 \end{bmatrix}$$

The static response of the structure is computed and plotted in Fig 6.3.9 by the commands

```
>> beam_spec = [40 5; 40 5; 40 5]; bc = [2 1];
>> constr = [1 1 0; 2 4 0];
>> setbeam2(beam_spec, bc, constr)
>> sd = [1 1 -0.08; 2 4 0.05];
>> beam2fbs(0, 0, sd)
```

TABLE 6.3.4 Specification of Support Settlement

Constraint Type		*Support Displacement*
(C1) Hinge		`SD_j = [node_no 1` u_c`]` where u_c is positive in the upward direction
(C2) Translational spring		`SD_j = [node_no 2` u_c`]` where u_c is positive in the upward direction
(C3) Torsional spring		`SD_j = [node_no 3` θ_c`]` where θ_c is given in radians and positive in the counterclockwise direction
(C4) Sliding constraint		`SD_j = [node_no 4` θ_c`]` where θ_c is given in radians and positive in the counterclockwise direction

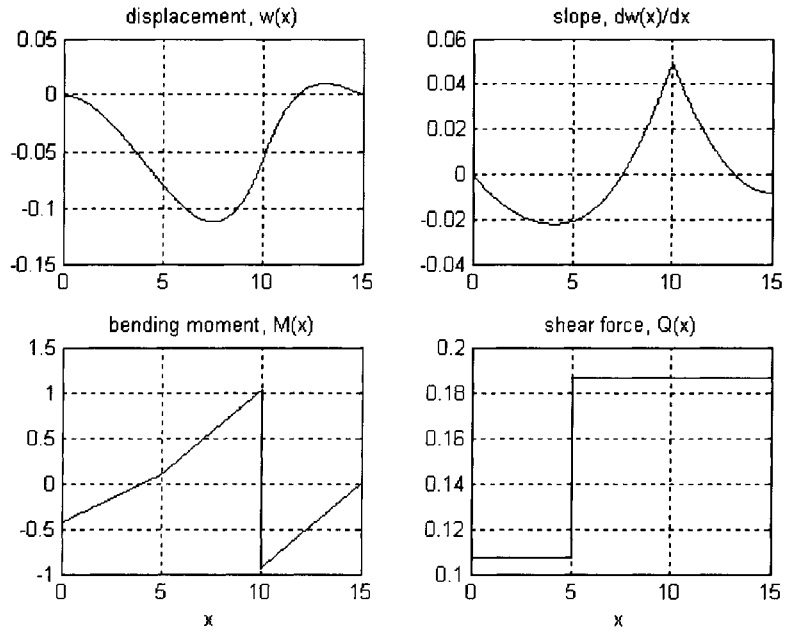


FIGURE 6.3.9

Also, the reactions at the supports are obtained as follows

```
Reactions at the Nodes of the Beam Structure
-------------------------------------------
At left boundary:   Mc = 0.43429,  Rc = 0.10697
                w0 = 0,  angle0 = 0 degrees

At node 1:          Mc = 0,  Rc = 0.079543
                W=-0.08, angle=-1.1949 degrees

At node 2:          Mc = 1.9657,  Rc = 0
                W=-0.055714, angle=2.8624 degrees

At right boundary:  Mc = 0,  Rc = -0.18651
                wL = 0,  angleL = -0.47473 degrees
```

6.3.5 Total Response

When a beam structure is subject to external loads, boundary disturbances, and support displacements simultaneously, its total static response can be obtained by function `beam2fbs`, as shown in Window 3.5. The Toolbox also has a function `plotbeam2` for plotting the beam response based on the results obtained by `beam2fbs`; see Window 3.6.

Window 3.5. Function `beam2fbs` for Total Response

MATLAB Function: **`beam2fbs`**

Purpose:
To compute the static response of a multispan beam subject to external and boundary disturbances, and support settlement.

Synopsis:

```
beam2fbs(Load_Spec, BD_Spec, SD_Spec)
[y, r] = beam2fbs(Load_Spec, BD_Spec, SD_Spec)
```

Description:
`beam2fbs(Load_Spec, BD_Spec, SD_Spec)` plots the total response of the beam structure, where `Load_Spec` is a matrix specifying external loads by Table 6.3.3, `BD_Spec` is a row vector specifying boundary disturbances by Table 6.3.1, and `SD_Spec` is a matrix specifying support settlement by Table 6.3.4. Any of `Load_Spec`, `BD_Spec`, `SD_Spec` can be replaced by 0 or `[ ]` if the corresponding disturbances do not exist. For instance, if a beam structure is subject to external loads and support settlement (but no boundary disturbances), its response can be plotted by

```
beam2fbs(Load_Spec, 0, SD_Spec).
```

`[y, r] = beam2fbs(Load_Spec, BD_Spec, SD_Spec)` returns the computed beam response in a three-dimensional matrix `y`, and the reactions and displacements at the nodes of the beam structure in a matrix `r`. With `y`, the beam response can be plotted by function `plotbeam2(y)`; see Window 3.6.

Window 3.6. Function `plotbeam2`

MATLAB Function: **`plotbeam2`**

Purpose:
To plot static response of a multispan beam structure.

Synopsis:

```
plotbeam2(y)
plotbeam2(y, opt)
[resp, x] = plotbeam2(y)
```

Description:
`plotbeam2(y)` plots spatial distributions of the response of a beam structure in four figures, namely, displacement $w(x)$, rotation $\theta(x) = \frac{d}{dx}w(x)$, bending moment $M(x)$, and shear force $Q(x)$. Here `y` is a matrix of the beam response data that is obtained by function `beam2fbs`.

`plotbeam2(y, opt)` plots the response of the beam structure with the following options:

`opt` = 1: plot displacement $w(x)$ only;
`opt` = 2: plot slope $\theta(x)$ only;
`opt` = 3: plot bending moment $M(x)$ only;
`opt` = 4: plot shear force $Q(x)$ only.

`[resp, x] = plotbeam2(y)` returns the computed response of the structure in a four-row matrix `resp` and the spatial points of computation in a vector `x`. Here, `resp(1,:)` stores displacement $w(x)$, `resp(2,:)` rotation $\theta(x)$, `resp(3,:)` bending moment $M(x)$, and `resp(4,:)` shear force $Q(x)$. With the output data `resp` and `x`, the response of the beam structure can be plotted through use of the MATLAB function `plot`. For instance, `plot(x, resp(3,:))` plots the spatial distribution of bending moment.

EXAMPLE 3.6

The beam structure in Fig. 6.3.10 is under a pointwise force ($q = -10$), a moment at its right end ($M_L = 3.0$), and a foundation deflection $u_c = -0.007$ at the elastic support. By the commands

```
>> setbeam2([60 1.5; 60 1; 25 1], [1 1], [1 2 50])
>> y = beam2fbs([1 0 0.72 -10 0 0 0], [0 0 0 3.0], [1 2 -0.007]);
>> plotbeam2(y, 3)
>> figure(2)
>> plotbeam2(y, 4)
```

the bending moment and shear force of the beam structure is plotted in Fig. 6.3.11.

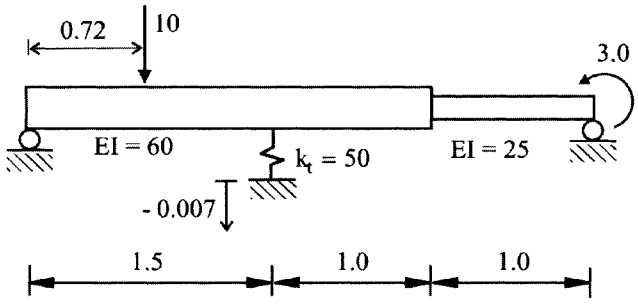


FIGURE 6.3.10

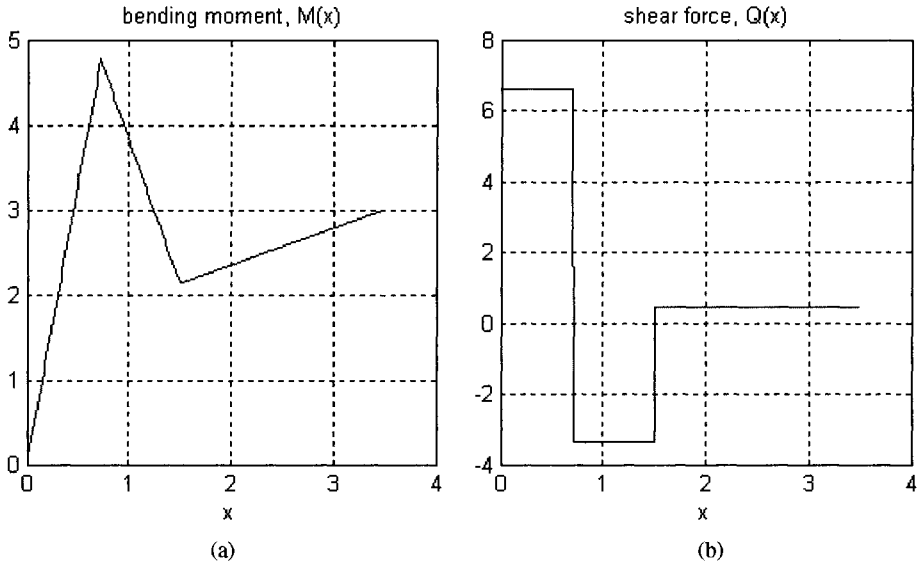


FIGURE 6.3.11

6.3.6 Influence Lines

The influence lines of a structure represent the variation of the response of the structure at a specific point or the reactions at a support as a concentrated load moves along the structure. Influence lines are useful in design of structures that are subject to moving forces or loadings of variable placement. Examples of such structures include highway and railroad bridges, industrial crane rails, and conveyors.

Construction of influence lines for a beam structure is illustrated in Fig. 6.3.12, where a *downward* unit force is placed at a variable location x_f along the structure. The response (bending moment, shear force, and deflection) of the structure at a specific point or the reactions at supports can be determined by static analysis. As the force travels along the structure, these response/reaction quantities are obtained as functions of x_f. One such function, when plotted versus the moving force location x_f, forms an influence line. For instance, for the structure in Fig. 6.3.12, the influence lines for the reaction force and moment at boundary A are $R_A(x_f)$ and $M_A(x_f)$; the influence line for the transverse reaction at support B is $R_B(x_f)$; and the influence

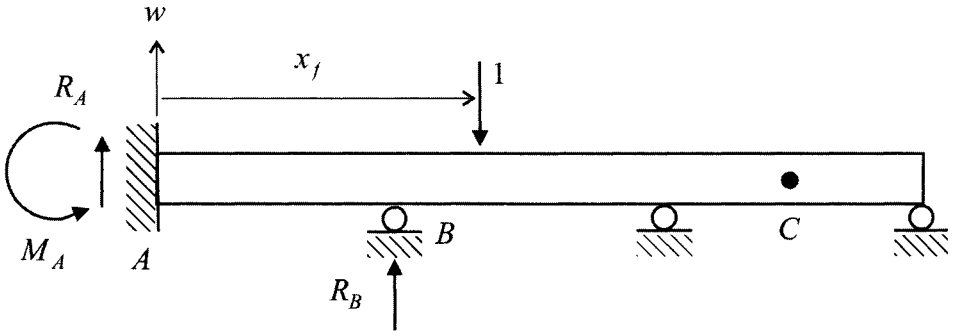


FIGURE 6.3.12 A beam structure under a unit load of variable location.

lines for bending moment, shear force, and deflection at point C are $M_C(x_f)$, $Q_C(x_f)$, and $w_C(x_f)$, respectively.

For statically determinate beam structures, the influence lines for reactions and internal forces are piecewise linear. For those structures, influence lines can be quickly constructed by using the Müller-Breslau principle or the virtual work principle, as discussed in many textbooks on structural mechanics [References (1) and (4)]. However, for a multispan beam structure that is statically indeterminate, the influence lines are usually curved, and they can be constructed only after the unknown reactions at the supports and boundaries are determined. Examples 3.7 and 3.8 show this difference.

The Toolbox of this chapter provides two MATLAB functions, `infline` and `inflreact`, for plotting influence lines for multispan beam structures; see Windows 3.6 and 3.7. One special feature of the functions is that they can be used to plot influence lines for both statically determinate and statically indeterminate beam structures.

Window 3.7. Function `infline`

MATLAB Function: **`infline`**

Purpose:
To plot influence lines for the response of an N-spanned beam structure at a selected point.

Synopsis:

```
infline(beam_no, xr)
infline(beam_no, xr, npts)
[y, xf] = infline(beam_no, xr, npts)
```

Description:
`infline(beam_no, xr)` plots the influence lines for the displacement, rotation, bending moment, and shear force of a beam structure of N elements at a selected point. Here `beam_no` is an integer between 1 and N identifying the beam element on which the point is selected and `xr` is the local coordinate of the point measured from the left end of the element. `Infline` also shows the maximum values of the influence line and locations.

Window 3.7. Function `infline` (*Continued*)

`infline(beam_no, xr, npts)` sets up the number `npts` of spatial points along the structure at which influence lines are computed and plotted. These spatial points actually are the locations of the traveling unit force shown in Fig. 6.3.12. By default, `npts` = 201.

`[y, xf] = infline(beam_no, xr, npts)` returns the computed data of influence lines in a four-row matrix `y` and the locations of the traveling unit force in a vector `xf`. Here `y(1,:)` stores displacement, `y(2,:)` rotation, `y(3,:)` bending moment, and `y(4,:)` shear force. With `y` and `xf`, the influence lines can be plotted through use of the MATLAB function `plot`. For instance, `plot(xf, y(3,:))` plots the influence line for the bending moment at point `xr`.

Window 3.8. Function `inflreact`

MATLAB Function: **`inflreact`**

Purpose:
To plot the influence lines of the reactions at a support of a beam structure.

Synopsis:

```
inflreact(node_no)
inflreact(node_no, npts)
[y, xf] = inflreact(node_no, npts)
```

Description:
`inflreact(node_no)` plots the influence lines of the reaction moment and force at a support of a beam structure where `node_no` is an integer between 0 and N identifying the node number of the support. `node_no = 0` identifies the left boundary of the structure while `node_no = N` identifies the right boundary. If no constraint is specified at the selected node or if only constraint C5 (hinge connection) is at the node, both the reaction moment and force are zero, and their influence lines should be zero lines. `inflreact` also shows the maximum values of the influence line and locations.

`inflreact(node_no, npts)` sets up the number `npts` of spatial points along the structure at which influence lines are computed and plotted. These spatial points actually are the locations of the traveling unit force shown in Fig. 6.3.12. By default, `npts` = 201.

`[y, xf] = inflreact(node_no, npts)` returns the computed data of influence lines in a two-row matrix `y`, and the locations of the traveling unit force in a vector `xf`. Here `y(1,:)` stores the reaction moment and `y(2,:)` the reaction force. With `y` and `xf`, the influence lines at the support can be plotted through use of the MATLAB function `plot`. For instance, `plot(xf, y(2,:))` plots the influence line for the reaction force at node `node_no`.

EXAMPLE 3.7

Consider the beam structure in Fig. 6.3.13, which is statically determinate. The influence line for the reaction force R_B at support B is obtained in Fig. 6.3.14 by the following MATLAB commands:

```
>> beam = [70 5; 70 5]; bc = [1 3]; constr = [1 1 0];
>> setbeam2(beam, bc, constr)
>> inflreact(1)
```

Furthermore, the influence lines for the response of the beam structure at point C are plotted in Fig. 6.3.15 by the command

```
>> infline(1, 2.5)
```

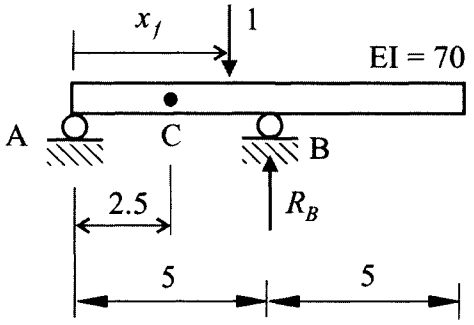


FIGURE 6.3.13

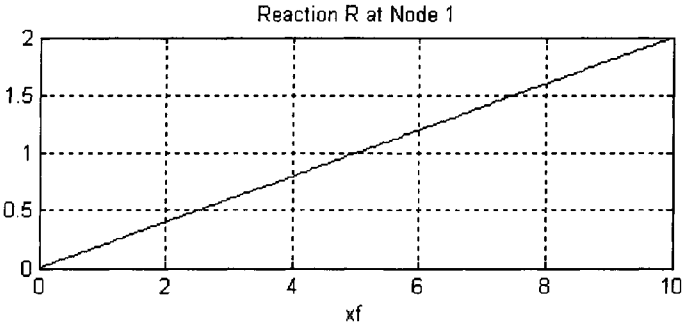


FIGURE 6.3.14

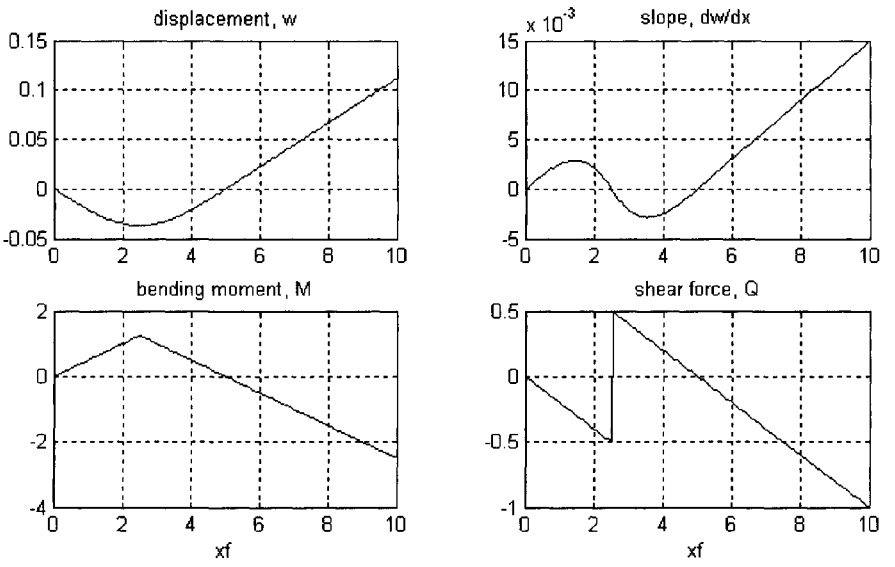


FIGURE 6.3.15

EXAMPLE 3.8

Consider the statically indeterminate beam structure in Fig. 6.3.16. The influence lines for the response of the beam structure at point C are plotted by

```
>> beam = [250 10; 250 10; 250 10];
>> bc = [2 1];
>> constr = [1 1 0; 2 1 0];
>> setbeam2(beam, bc, constr)
>> infline(3, 7)
```

which yields Fig. 6.3.17 and the following information

```
Maximum displacement w_max = -0.049593 when the unit load is at
   xf = 25.8209
Maximum bending moment M_max = 1.8699 when the unit load is at
   xf = 27.0149
```

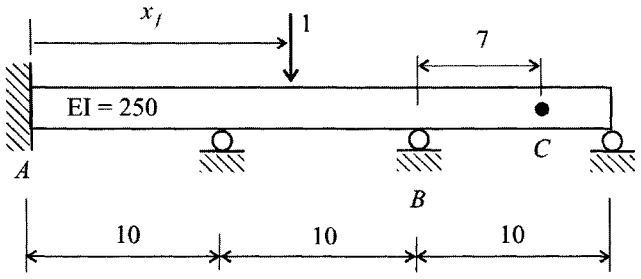


FIGURE 6.3.16

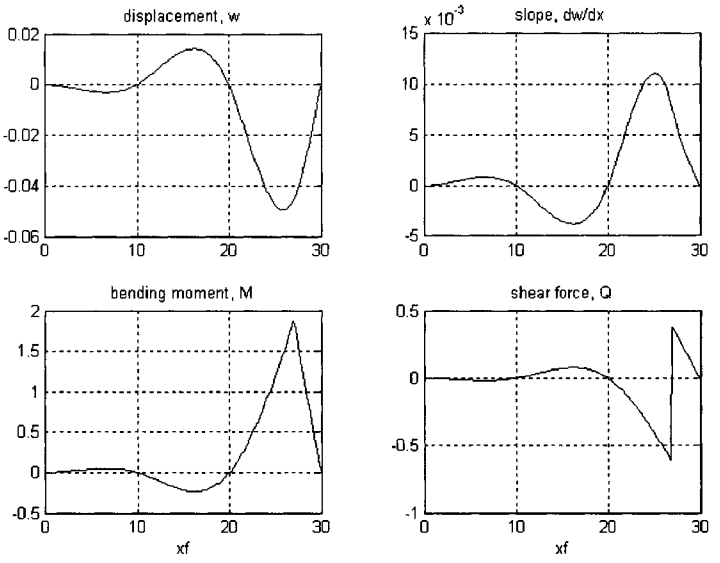


FIGURE 6.3.17 Influence lines at point *C*.

```
Maximum shear force Q_max = -0.61047 when the unit load is at
   xf = 26.8657
Here xf is a global coordinate measured from the left end of
   the structure
```

The influence lines for the reactions at boundary A (Node 0) and support B (Node 2) are also constructed by

```
>> inflreact(0)
>> inflreact(2)
```

which yields Figs. 6.3.18 and 6.3.19. As observed, the influence lines for the reactions and internal forces in Example 3.7 (a statically determinate structure) are piecewise

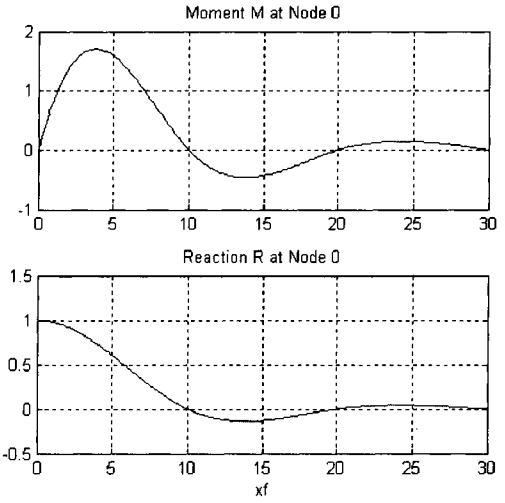


FIGURE 6.3.18 Influence lines at boundary *A*.

linear, i.e., linear in segments; the influence lines for the reactions and internal forces in Example 3.8 (a statically indeterminate structure) are curved.

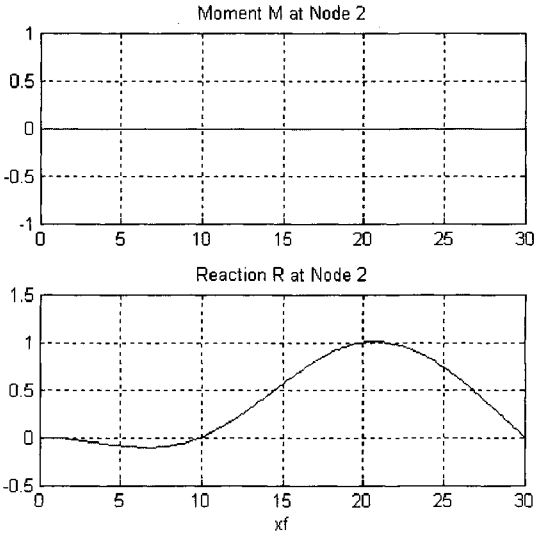


FIGURE 6.3.19 Influence lines at support B.

6.4 The Distributed Transfer Function Method

This section presents an exact analytical solution method, namely, the Distributed Transfer Function Method (DTFM), which is used to develop the MATLAB functions of the Toolbox for this chapter. For more information on the DTFM, refer to Appendix C and Reference (5). The reader who is only interested in using the Toolbox can skip this section.

State Formulation for Beam Elements

In the DTFM-based analysis of a beam structure shown in Fig. 6.2.1, the governing equation (6.1) for the ith element is cast in spatial state form

$$\frac{d}{dx}\{\eta(x)\} = [F]\{\eta(x)\} + \frac{1}{EI_i} f(x)\{e_4\}, \quad 0 \le x \le l_i \tag{6.4}$$

with

$$\{\eta(x)\} = \begin{pmatrix} w(x) \\ w'(x) \\ w''(x) \\ w'''(x) \end{pmatrix}, \qquad [F] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.5}$$

where $\{e_4\} = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T$, and $w' = dw/dx$. The boundary conditions of the beam element at nodes i-1 and i are described by

$$[M_e]\{\eta(0)\} + [N_e]\{\eta(l_i)\} = \{u_i\} \tag{6.6}$$

where

$$[M_e] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad [N_e] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \qquad \{u_i\} = \begin{pmatrix} w_{i-1} \\ \theta_{i-1} \\ w_i \\ \theta_i \end{pmatrix} \tag{6.7}$$

with w_i and θ_i being the transverse displacement and rotation of the element at node i, respectively. For the first element

$$[M_e] = [M_b], \quad \{u_1\} = \begin{pmatrix} 0 \\ 0 \\ w_1 \\ \theta_1 \end{pmatrix}$$

and for the last (Nth) element

$$[N_e] = [N_b], \quad \{u_N\} = \begin{pmatrix} w_{N-1} \\ \theta_{N-1} \\ 0 \\ 0 \end{pmatrix}$$

where matrices $[M_b]$ and $[N_b]$ are formed by the boundary conditions given in Table 6.2.1. For instance, for a clamped-free (cantilever) beam structure

$$[M_b] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad [N_b] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & EI & 0 \\ 0 & 0 & 0 & -EI \end{bmatrix}.$$

Element Stiffness Matrix

The solution of Eq. (6.4) with the boundary conditions (6.6) is

$$\{\eta(x)\} = \frac{1}{EI_i} \int_0^L G_i(x,\xi) f(\xi) d\xi \, \{e_4\} + H_i(x) \{u_i\}, \quad 0 \le x \le l_i \tag{6.8}$$

where the matrix Green's functions (Reference 5)

$$G_i(x,\xi) = \begin{cases} H_i(x)\,[M_e]\, e^{-[F]\xi} & \text{for } x \ge \xi \\ -H_i(x)\,[N_e]\, e^{[F](l_i-\xi)} & \text{for } x \le \xi \end{cases} \tag{6.9}$$

$$H_i(x) = e^{[F]x} \left([M_e] + [N_e]\, e^{[F]l_i} \right)^{-1}$$

The exponential matrix $e^{[F]x}$ has the form

$$e^{[F]x} = \begin{bmatrix} 1 & x & \frac{1}{2}x^2 & \frac{1}{6}x^3 \\ 0 & 1 & x & \frac{1}{2}x^2 \\ 0 & 0 & 1 & x \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

Thus, the Green's functions in Eq. (6.9) are in exact and closed form.

The internal forces of the ith beam element are expressed as

$$\begin{pmatrix} M(x) \\ Q(x) \end{pmatrix} = EI_i \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \{\eta(x)\} \tag{6.11}$$

Write

$$G_i(x,\xi) = \left[g_{jk}^{(i)}(x,\xi)\right], \quad H_i(x) = \left[h_{jk}^{(i)}(x)\right] \tag{6.12}$$

Substitute Eq. (6.8) into Eq. (6.11) to obtain force-displacement relation of the element

$$\{p_i\} = [K_i]\{u_i\} + \{q_i\} \tag{6.13}$$

where $\{p_i\}$ is the vector of nodal forces applied to the ends of the ith element, as shown in Fig. 6.4.1, $\{u_i\}$ is the vector of nodal displacements, $[K_i]$ is a four-by-four stiffness matrix of the element, and $\{q_i\}$ is a vector of transmitted nodal forces due to external load applied to the interior of the element. It can be shown that

$$\{p_i\} = \begin{pmatrix} V_{i-1}^{(i)} \\ M_{i-1}^{(i)} \\ V_i^{(i)} \\ M_i^{(i)} \end{pmatrix} = EI_i \begin{pmatrix} w'''(0) \\ -w''(0) \\ -w'''(l_i) \\ w''(l_i) \end{pmatrix}, \quad \{u_i\} = \begin{pmatrix} u_{i-1} \\ \theta_{i-1} \\ u_i \\ \theta_i \end{pmatrix} = \begin{pmatrix} w(0) \\ w'(0) \\ w(l_i) \\ w'(l_i) \end{pmatrix} \tag{6.14a}$$

$$[K_i] = EI_i \begin{bmatrix} h_{41}^{(i)}(0) & h_{42}^{(i)}(0) & h_{43}^{(i)}(0) & h_{44}^{(i)}(0) \\ -h_{31}^{(i)}(0) & -h_{32}^{(i)}(0) & -h_{33}^{(i)}(0) & -h_{34}^{(i)}(0) \\ -h_{41}^{(i)}(l_i) & -h_{42}^{(i)}(l_i) & -h_{43}^{(i)}(l_i) & -h_{44}^{(i)}(l_i) \\ h_{31}^{(i)}(l_i) & h_{32}^{(i)}(l_i) & h_{33}^{(i)}(l_i) & h_{34}^{(i)}(l_i) \end{bmatrix} \tag{6.14b}$$

$$\{q_i\} = \int_0^{l_i} \begin{pmatrix} g_{44}^{(i)}(0,\xi) & -g_{34}^{(i)}(0,\xi) & -g_{44}^{(i)}(l_i,\xi) & g_{34}^{(i)}(l_i,\xi) \end{pmatrix}^T f(\xi)d\xi \tag{6.14c}$$

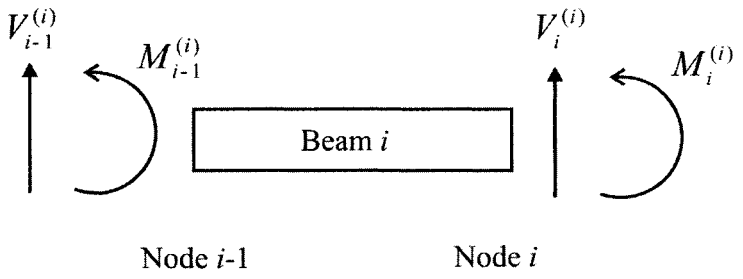


FIGURE 6.4.1 The nodal forces of an element.

Here, $V_{i-1}^{(i)}$ and $M_{i-1}^{(i)}$ are the nodal force and moment applied to node i-1, $V_i^{(i)}$ and $M_i^{(i)}$ are the nodal force and moment applied to node i, and $g_{jk}^{(i)}$ and $h_{jk}^{(i)}$ are the components of the Green's functions. By Eq. (6.9), the integrals in Eq. (6.14c) can be obtained by exact quadrature for general forcing functions.

Assembly of Beam Structure

The beam structure is assembled from its elements through force balance at all nodes. Constraints described in Section 6.2 are also properly introduced in the structure assembly. This eventually leads to a global equilibrium equation

$$[K_g]\{U\} = \{F\} \tag{6.15}$$

where $\{U\}$ is a vector of nodal displacements, $\{F\}$ is a vector consisting of external loads and transmitted nodal forces, and $[K_g]$ is the global stiffness matrix that is assembled from the element stiffness matrices $[K_i]$.

Exact Static Response

The static response of the beam structure is obtained by first solving Eq. (6.15) for the nodal displacements

$$\{U\} = [K_g]^{-1}\{F\} \tag{6.16}$$

and then substituting the result into Eq. (6.8). During this process, no approximation or discretization is made. The above-mentioned DTFM analysis is extremely convenient in computer coding, and it delivers exact solutions for multispan beam structures subject to general external loads, boundary disturbances, and support settlement.

EXAMPLE 4.1

The DTFM is illustrated on the two-span beam in Fig. 6.4.2, which is subject to a spring constraint (k) and a pointwise force (q). The response of the beam elements, by Eq. (6.8), is obtained as follows:

For Beam 1

$$\{\eta(x)\} = H_1(x)\begin{pmatrix} 0 \\ 0 \\ w_1 \\ \theta_1 \end{pmatrix}, \quad 0 \le x \le l_1 \tag{6.17a}$$

For Beam 2

$$\{\eta(x)\} = \frac{q}{EI_2}\begin{pmatrix} g_{14}^{(2)}(x,\Delta) \\ g_{24}^{(2)}(x,\Delta) \\ g_{34}^{(2)}(x,\Delta) \\ g_{44}^{(2)}(x,\Delta) \end{pmatrix} + H_2(x)\begin{pmatrix} w_1 \\ \theta_1 \\ 0 \\ 0 \end{pmatrix}, \quad 0 \le x \le l_2 \tag{6.17b}$$

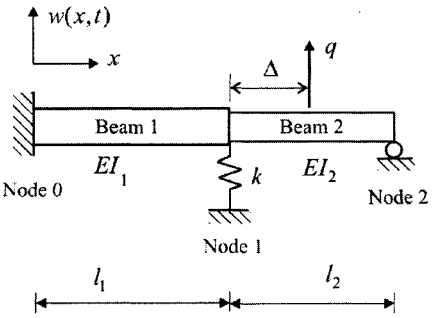


FIGURE 6.4.2 A two-span beam.

where w_1 and θ_1 are the transverse displacement and rotation of the beam structure at node 1. Force balance at node 1, which is shown in Fig. 6.4.3, leads to the following equations

$$\begin{aligned} M_1^{(1)} + M_1^{(2)} &= 0 \\ V_1^{(1)} + V_1^{(2)} + f_s &= 0 \end{aligned} \tag{6.18}$$

where $f_s = kw_1$ is the constraint force exerted to node 1 by the spring. Substituting Eqs. (6.17) into Eqs. (6.18) yields the equilibrium equation

$$\begin{bmatrix} -EI_1 h_{43}^{(1)}(l_1) + EI_2 h_{41}^{(1)}(0) + k & -EI_1 h_{44}^{(1)}(l_1) + EI_2 h_{42}^{(1)}(0) \\ EI_1 h_{33}^{(1)}(l_1) - EI_2 h_{31}^{(1)}(0) & EI_1 h_{34}^{(1)}(l_1) - EI_2 h_{32}^{(1)}(0) \end{bmatrix} \begin{pmatrix} w_1 \\ \theta_1 \end{pmatrix} = q \cdot \begin{pmatrix} -g_{44}^{(2)}(0,\Delta) \\ g_{34}^{(2)}(0,\Delta) \end{pmatrix} \tag{6.19}$$

Solution of Eq. (6.19) for the nodal displacements and substitution of the result into Eqs. (6.17) gives the response of the structure.

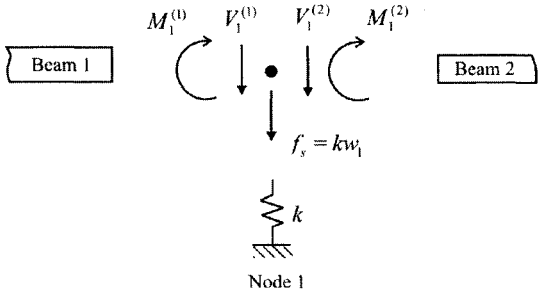


FIGURE 6.4.3 Force balance at node 1.

6.5 Quick Solution Guide

System Description

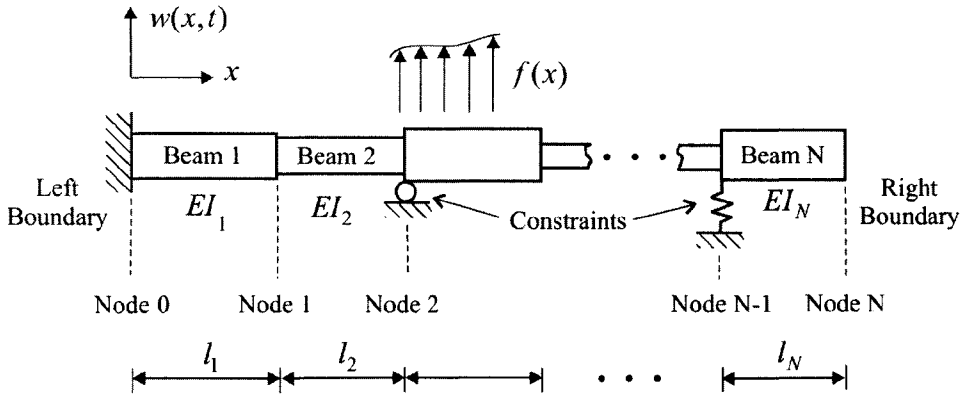


FIGURE 6.5.1 Schematic of a constrained multispan beam.

Nodes:

Nodes 0 and N: left and right boundaries

Nodes 1, 2, . . . , N-1: interior nodes at which adjacent beam elements are interconnected, and at which constraints or supports may be imposed

Euler-Bernoulli Beam Model of the ith Element:

$$EI_i \frac{d^4}{dx^4} w(x) = f(x), \quad 0 \le x \le l_i$$

Static response:

Displacement (transverse deflection): $w(x)$

Slope or rotation: $\theta(x) = \dfrac{d}{dx} w(x)$

Bending moment: $M(x) = EI_i \dfrac{d^2}{dx^2} w(x)$

Shear force: $Q(x) = EI_i \dfrac{d^3}{dx^3} w(x)$

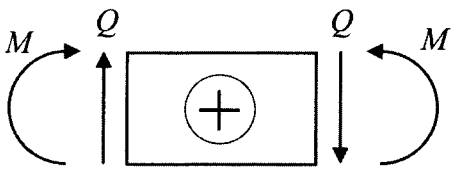


FIGURE 6.5.2 Sign convention of bending moment and shear force.

Problems to Solve

This Toolbox provides functions for finding solutions of the following problems:

(a) Static response to external loads;
(b) Static response to boundary disturbances;
(c) Static response to settlement of supports; and
(d) Influence lines for reactions at supports and internal forces.

MATLAB Functions

This chapter contains a toolbox of MATLAB functions for static analysis of constrained, multispan Euler-Bernoulli beams; see the table below. Refer to the corresponding window or section for the utility of each function. The application of some major functions is summarized in the following pages.

	System Setup	
`setbeam2`	Set up beam parameters, boundary conditions, and constraints	Window 3.1
	Static Analysis of Beams	
	Compute beam response subject to external loads	Window 3.2
	Compute beam response due to boundary disturbances	Window 3.3
`beam2fbs`	Compute beam response due to support settlements	Window 3.4
	Compute total response due to external and boundary disturbances and support settlement	Window 3.5
`plotbeam2`	Plot spatial distribution of beam response	Window 3.6
`infline`	Plot influence lines for response at a specific point	Window 3.7
`inflreact`	Plot influence lines for reactions at a support or node	Window 3.8
	Utilities	
`systinfo`	Display the information of a multispan beam	Section 6.3.1
`TBdemo`	Show how the Toolbox works and what it can do	Section 6.1
`TBinfo`	Show the information of the Toolbox	Section 6.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 6.1

Solution Procedure

Static analysis of a constrained multispan beam structure takes the following two steps:

Step 1. Set up system parameters, boundary conditions, and constraints

```
>> setbeam2(Beam_Spec, BC_Spec, Constr_Spec)
```

Step 2. Compute static response

To compute response subject to external forces, type

```
>> beam2fbs(Load_Spec)
```

To compute response subject to boundary disturbances, type

```
>> beam2fbs(0, BD_Spec)
```

To compute response subject to settlement of supports, type

```
>> beam2fbs(0, 0, SD_Spec)
```

For total response, type

```
>> beam2fbs(Load_Spec, BD_Spec, SD_Spec)
```

To plot influence lines, type

```
>> infline(beam_no, xr)
```

Note. `Beam_Spec`, `BC_Spec`, and `Constr_Spec` specify the parameters, boundary conditions, and constraints of the beam structure, respectively. `Load_Spec` specifies external loads; `BD_Spec` specifies boundary disturbances; and `SD_Spec` specifies support settlement. Specification of these arguments is given as follows.

Specification of Parameters of Beam Elements

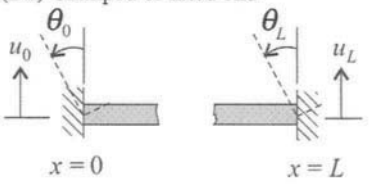
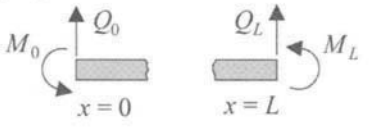
$$\texttt{Beam\_Spec} = \begin{bmatrix} EI_1 & l_1 \\ EI_2 & l_2 \\ \vdots & \vdots \\ EI_N & l_N \end{bmatrix}$$

where the ith row of the matrix contains the bending stiffness and length of the ith beam element.

Specification of Boundary Conditions and Boundary Disturbances

Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`

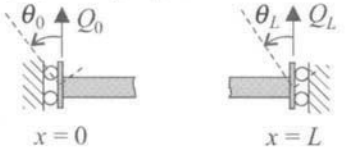
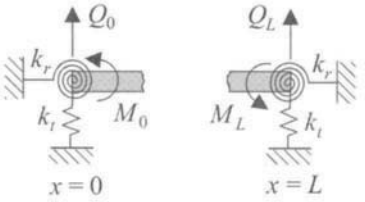
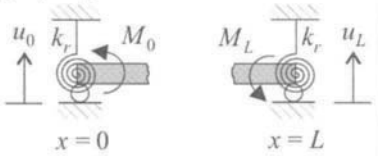
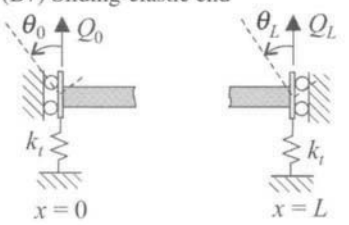
Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

Boundary Condition	*Left Boundary (x = 0)*		*Right Boundary (x = L)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B1) Pinned or hinged end	[1]	$[u_0 \quad M_0]$	[1]	$[u_L \quad M_L]$
(B2) Clamped or fixed end 	[2]	$[u_0 \quad \theta_0]$	[2]	$[u_L \quad \theta_L]$
(B3) Free end 	[3]	$[M_0 \quad Q_0]$	[3]	$[M_L \quad Q_L]$

(continued)

Boundary Conditions: `BC_Spec = [BC_Left BC_Right]`

Boundary Disturbances: `BD_Spec = [BD_Left BD_Right]`

Boundary Condition	*Left Boundary (x = 0)*		*Right Boundary (x = L)*	
	BC_Left	***BD_Left***	***BC_Right***	***BD_Right***
(B4) Sliding or guided end 	[4]	$[\theta_0 \quad Q_0]$	[4]	$[\theta_L \quad Q_L]$
(B5) Elastic end 	[5 k_r k_t]	$[M_0 \quad Q_0]$	[5 k_r k_t]	$[M_L \quad Q_L]$
(B6) Hinged-elastic end 	[6 k_r]	$[u_0 \quad M_0]$	[6 k_r]	$[u_L \quad M_L]$
(B7) Sliding-elastic end 	[7 k_t]	$[\theta_0 \quad Q_0]$	[7 k_t]	$[\theta_L \quad Q_L]$

u_0, u_L—end displacement; θ_0, θ_L—end rotation in radians;
M_0, M_L—end moment; Q_0, Q_L—end transverse force;
k_r—torsional spring coefficient; k_t—translational spring coefficient.
Positive direction of end displacement and transverse force: `upward`
Positive direction of end rotation and moment: `counterclockwise`

The above table is the same as Table 6.3.1.

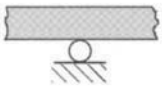
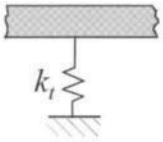
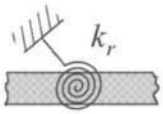
Specification of Constraints

$$\texttt{Constr\_Spec} = \begin{bmatrix} \texttt{Constr\_1} \\ \texttt{Constr\_2} \\ \vdots \\ \texttt{Constr\_Nc} \end{bmatrix}$$

The *j*th row of the matrix specifies the *j*th constraint, and is of the format

```
Constr_j = [node_no constr_type_no p]
```

where `node_no` is node number (1, 2, . . . , N-1), `constr_type_no` is an integer between 1 and 5 specifying one of the five types of constraints (C1 to C5) shown in the following table (the same as Table 6.3.2), and `p` is a spring coefficient for constraints C2 and C3, and zero for others. If the beam structure has no constraints, set `Constr_Spec = 0` or `Constr_Spec = [ ]`.

Constraint Type		`Constr_j`
(C1) Hinge		`[node_no 1 0]`
(C2) Translational spring		`[node_no 2` k_t`]`
(C3) Torsional spring		`[node_no 3` k_r`]`
(C4) Sliding constraint		`[node_no 4 0]`
(C5) Hinge connection		`[node_no 5 0]`

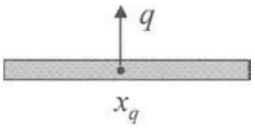
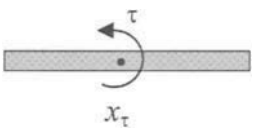
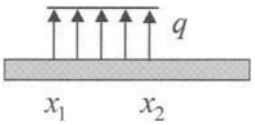
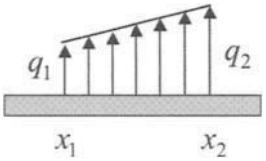
Specification of External Loads

$$\texttt{Load\_Spec} = \begin{bmatrix} \texttt{Load\_1} \\ \texttt{Load\_2} \\ \vdots \\ \texttt{Load\_Nf} \end{bmatrix}$$

where each row of the matrix specifies an external load and `Nf` is the total number of loads. A typical row is of the form

```
Load_j = [beam_no load_type p1 p2 p3 p4 p5]
```

where beam_no is an element number, load_type is an integer between −1 to 3 identifying one of the five types of external loads given in the following table (the same as Table 6.3.3), and p1 to p5 are parameters describing the location and form of the load.

(L1) Pointwise force  $f(x) = q\delta(x - x_q)$ `Load_j = [beam_no 0` x_q `q 0 0 0]`	(L2) Torque:  $f(x) = -\tau\frac{d}{dx}\delta(x - x_\tau)$ `Load_j = [beam_no -1` x_τ τ `0 0 0]`
(D1) Uniform distribution  $f(x) = q \quad \text{for } x_1 \le x \le x_2$ `Load_j = [beam_no 1` x_1 x_2 `q 0 0 0]`	(D2) Linear distribution  $f(x) = q_1 + \frac{q_2 - q_1}{a}(x - x_1) \quad \text{for } x_1 \le x \le x_2$ where $a = x_2 - x_1$ `Load_j = [beam_no 2` x_1 x_2 q_1 q_2 `0]`
(D3) Quadratic distribution $f(x) = c_0 + c_1 x + c_2 x^2 \quad \text{for } x_1 \le x \le x_2$ `Load_j = [beam_no 3` x_1 x_2 c_0 c_1 c_2`]`	**Sign Convention** The positive direction of transverse loads (L1, D1 to D3) is upward. The positive direction of a torque (L2) is counterclockwise.

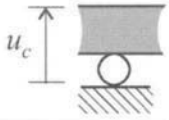
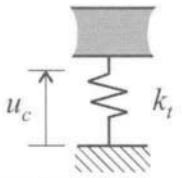
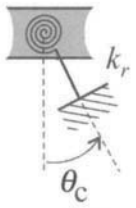
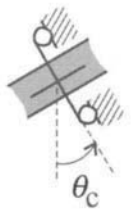
Specification of Displacements at Supports (settlement of supports)

$$\texttt{SD\_Spec} = \begin{bmatrix} \texttt{SD\_1} \\ \texttt{SD\_2} \\ \vdots \\ \texttt{SD\_Ns} \end{bmatrix}$$

where each row of the matrix specifies a support displacement and Ns is the total number of support settlement. A typical row has the form

```
SD_j = [node_no constr_type_no p]
```

where node_no is node number, constr_type_no is an integer between 1 and 4 specifying one of the four types of constraints given in the following table (the same as Table 6.3.4), and p is the specified support displacement or rotation.

Constraint Type	Support Displacement
(C1) Hinge 	`SD_j = [node_no 1 uc]` where u_c is positive in the upward direction
(C2) Translational spring 	`SD_j = [node_no 2 uc]` where u_c is positive in the upward direction
(C3) Torsional spring 	`SD_j = [node_no 3 θc]` where θ_c is given in radians and positive in the counterclockwise direction
(C4) Sliding constraint 	`SD_j = [node_no 4 θc]` where θ_c is given in radians and positive in the counterclockwise direction

6.6 References

1. Hibbeler R C 2001 *Structural Analysis*, 5th edition, Prentice-Hall: Upper Saddle River, New Jersey.
2. Kassimali A 1999 *Matrix Analysis of Structures*, Brooks Cole.
3. Przemienieck J S 1985 *Theory of Matrix Structural Analysis*, Dover Publications.
4. West H H and Geschwindner L F 2002 *Fundamentals of Structural Analysis*, 2nd edition, Wiley Text Books.
5. Yang B 1994 "Distributed Transfer Function Analysis of Complex Distributed Parameter Systems," *ASME Journal of Applied Mechanics*, Vol. 61: No. 1, pp. 84–92.

7 Static Analysis of Plane Trusses

Inside
- Getting Started
- System Description
- Solution by the Toolbox
- The Stiffness Method
- Quick Solution Guide
- References

7.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for static analysis of plane trusses.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

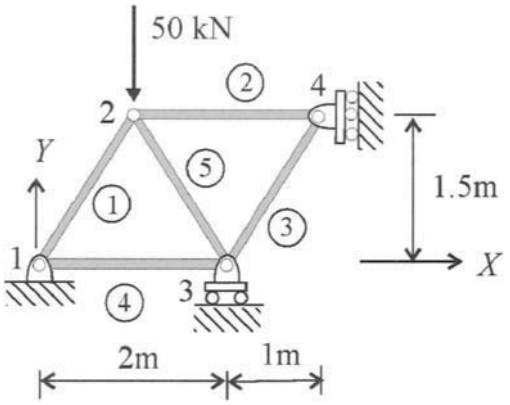


FIGURE 7.1.1 A truss under a vertical load at node 2.

Quick Tutorial

To show how to use the Toolbox, consider a truss in Fig. 7.1.1, where member identification numbers (in circles) and node identification numbers are assigned. The truss is subject to a vertical external load (50 kN) at node 2. There are three supports: a hinge at node 1, and one roller at each of nodes 3 and 4. Let all the members have the same stiffness (longitudinal rigidity) $EA = 850{,}000$ Pa. To obtain the static response of the truss, in the MATLAB command window, type:

```
>> EA = 850e3;
>> members = [EA 1 2; EA 2 4; EA 3 4; EA 1 3; EA 2 3];
>> nodes = [0 0; 1 1.5; 2 0; 3 1.5];
>> supports = [1 1 0; 3 2 0; 4 3 0];
>> settruss(members, nodes, supports);
>> load = [2 0 -50e3];
>> trussresp(load)
```

where >> is the prompt in the MATLAB command window. This yields the truss configurations (undeformed and deformed) plotted in Fig. 7.1.2, and the results given below:

```
Nodal displacements U & V in global coordinates X, Y
   node 1: U = 0, V = 0
   node 2: U = 0.0072227, V = -0.088457
   node 3: U = 0.035604, V = 0
   node 4: U = 0, V = 0.023736

Internal forces of members
   Member No.   Axial Force
       1        -32813.2008
       2        -3069.6445
       3        0
       4        15131.8444
       5        -27279.3205
```

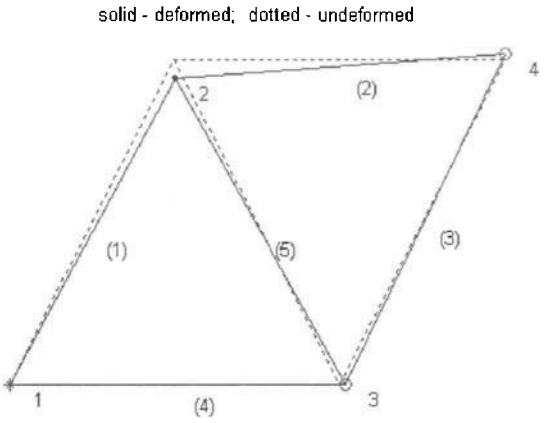


FIGURE 7.1.2 Truss configurations.

```
Reactions
   at node 1: Rx = 3069.6445
   at node 1: Ry = 27302.2334
   at node 3: Ry = 22697.7666
   at node 4: Rx = -3069.6445
```

In the above static analysis of the truss, two functions from the Toolbox are called:

(a) Function **settruss** that inputs the truss parameters and sets up the supports.
(b) Function **trussresp** that computes the response of the truss under the external load and displays the truss configurations.

These functions, along with others from the Toolbox, will be described in the subsequent sections.

For a quick solution, the reader may refer directly to the Quick Solution Guide (Section 7.5), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To better understand how the toolbox works, the reader is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 7.6.

7.2 System Description

Trusses are used to support roofs and bridges. A truss is an assembly of slender members that are pin-connected together at their ends; see Fig. 7.2.1 for an example. The joints where members are interconnected are called *nodes*. External loads are only applied to the nodes. As such, a truss member only undergoes longitudinal deformation (tension or compression) along its axis. In this chapter, trusses lying and deforming in a plane are modeled and analyzed.

Member Model

A member is modeled as a longitudinal bar that is subject to external loads only at its two ends; see Fig. 7.2.2. The longitudinal displacement of the bar is governed by the differential

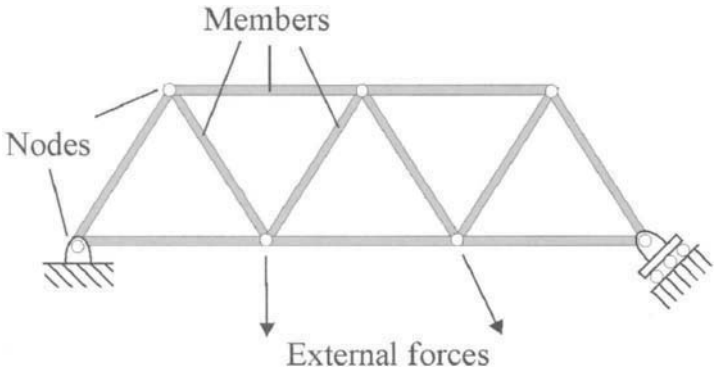


FIGURE 7.2.1 Schematic of a truss.

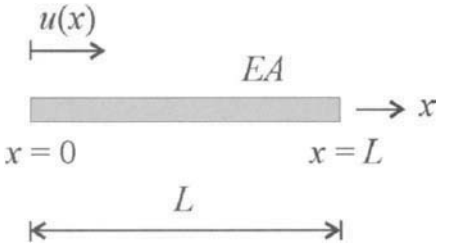


FIGURE 7.2.2 Truss member.

equation

$$EA\frac{d^2}{dx^2}u(x) = 0, \quad 0 \le x \le L \tag{7.1}$$

where EA and L are the stiffness (longitudinal rigidity) and length of the member, respectively, and x is the local coordinate of the member. The internal (axial) force of the member is

$$S(x) = EA\frac{d}{dx}u(x) \tag{7.2}$$

The displacement of the bar, which is the solution of Eq. (7.1), is of the form

$$u(x) = a_0 + a_1 x \tag{7.3}$$

where constants a_0 and a_1 are determined by the boundary conditions of the bar. Accordingly, the axial force of the bar is constant, i.e.,

$$S(x) = EAa_1 \tag{7.4}$$

Supports

The supports of a truss are placed at some of its nodes. Four types of supports for truss structures are shown in Fig. 7.2.3, where R_x, R_y, and R_n are support reactions. The description of these supports is as follows:

(S1) *Pin or hinge.* The displacements U and V of a truss at the support are both zero.

(S2) *Roller on a horizontal surface.* The vertical displacement V of a truss at the support is zero; the roller can move horizontally.

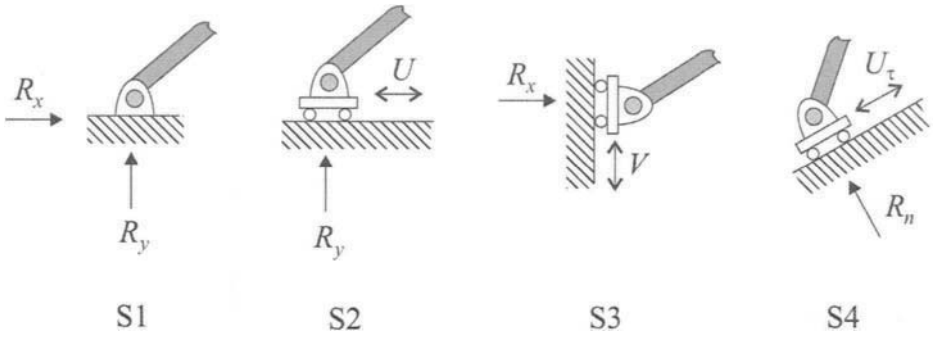


FIGURE 7.2.3 Four types of supports.

(S3) *Roller on a vertical surface.* The horizontal displacement U of a truss at the support is zero; the roller can move vertically.

(S4) *Roller on an incline.* The displacement V_n of a truss in the normal direction of the incline surface is zero. The roller can move tangentially along the incline surface.

Disturbances

The following three types of static disturbances are considered:

External forces—concentrated (pointwise) external forces applied to nodes;

Settlement of supports—deflections of supports in the directions of reactions; and

Fabrication errors and thermal effects—an increase or decrease of member length due to fabrication errors and thermal changes.

Static Problem and Solution

A fundamental problem for a plane truss is to determine its static response (displacements at nodes, internal forces of members, and reactions at supports) subject to external loads, support settlement, fabrication errors, and thermal changes. This chapter provides a toolbox of MATLAB functions for obtaining the solution of the problem; see Section 7.3. The solution method used by the Toolbox is presented in Section 7.4.

7.3 Solution by the Toolbox

The Toolbox of this chapter has several MATLAB functions for determining the static response of a truss structure. The solution procedure takes the following three steps:

Step 1. Set up the truss structure by function **`settruss`**;
Step 2. Determine the truss response by function **`trussresp`**; and
Step 3. Plot truss configurations by function **`plottruss`**.

These functions are presented in sequel.

7.3.1 System Setup

To model a truss structure by the Toolbox, the first thing is to label members and nodes with identification numbers. For instance, for a truss in Fig. 7.3.1, seven member numbers (in circles) and 5 node members $(1, 2, \ldots, 5)$ are assigned. In general, for a truss of M members

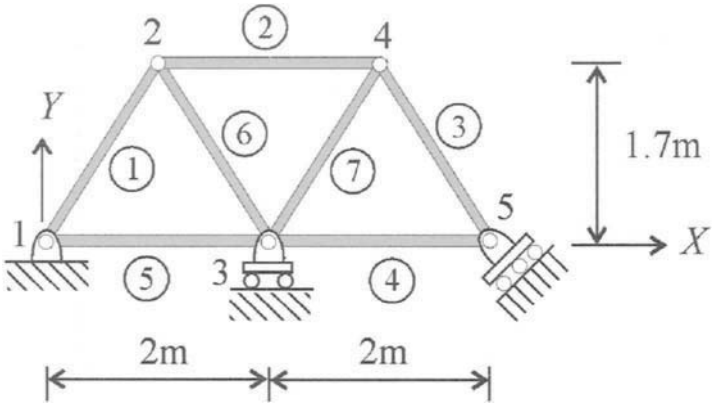


FIGURE 7.3.1

and N nodes, M member numbers and N node numbers will be assigned. These numbers are assigned in an arbitrary order.

The location of a node is described by a pair of global coordinates X and Y. For the truss in Fig. 7.3.1, the coordinates of the nodes are as follows:

$$\text{node 1 } (X = 0,\ Y = 0),\ \text{node 2 } (1, 1.7),\ \text{node 3 } (2, 0),\ \text{node 4 } (3, 1.7),\ \text{node 5 } (4, 0).$$

Once the node locations are expressed by global coordinates, the location, length, and orientation of each member are also confirmed because each member is bounded by two nodes. In truss analysis, the displacements at a node shall be called *nodal displacements*.

With the member and node identification numbers and node coordinates, function `settruss` from the Toolbox is called to set up the structure; see Window 3.1.

Window 3.1. Function settruss

MATLAB Function: **settruss**

Purpose:
To set up a truss structure.

Synopsis:

```
settruss(Member_Spec, Node_Spec, Support_Spec)
```

Description:
`settruss(Member_Spec, Node_Spec, Support_Spec)` undertakes the following tasks:

(a) It inputs the stiffness and node numbers of truss members by a matrix `Member_Spec`;
(b) It inputs the coordinates of all nodes by a matrix `Node_Spec`;
(c) It specifies the supports of the truss by a matrix `Support_Spec`; and
(d) It plots the undeformed configuration of the truss.

This function must be called before functions `trussresp` and `plottruss` can be used.

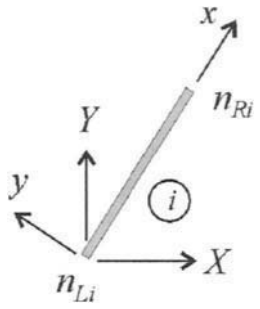


FIGURE 7.3.2 The local coordinates of a truss member.

In Window 3.1, function `settruss` specifies a truss structure by the following matrices:

(i) The members of the truss are specified by a matrix

$$\texttt{Member\_Spec} = \begin{bmatrix} EA_1 & n_{L1} & n_{R1} \\ EA_2 & n_{L2} & n_{L2} \\ \vdots & \vdots & \vdots \\ EA_M & n_{LM} & n_{LM} \end{bmatrix} \tag{7.5}$$

where in the ith row, EA_i is the stiffness (longitudinal rigidity) of the ith member, and n_{Li} and n_{Ri} are the node numbers of the left and right ends of the member, respectively. The node numbers also define a local coordinate system (x, y), with the positive x-axis oriented from node n_{Li} to node n_{Ri}; see Fig. 7.3.2.

(ii) The global coordinates of all the nodes are specified by a matrix

$$\texttt{Node\_Spec} = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ \vdots & \vdots \\ X_N & Y_N \end{bmatrix} \tag{7.6}$$

where X_i and Y_i are the coordinates of the ith node.

(iii) The supports of the truss are specified by a matrix

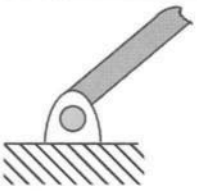
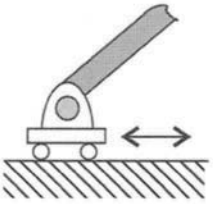
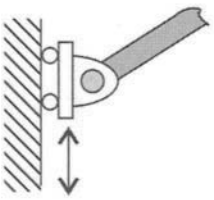
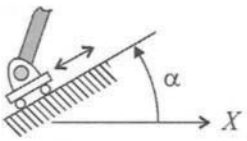
$$\texttt{Support\_Spec} = \begin{bmatrix} \texttt{Support\_1} \\ \texttt{Support\_2} \\ \vdots \\ \texttt{Support\_Ns} \end{bmatrix} \tag{7.7}$$

where `Ns` is the total number of supports. The kth row of the matrix specifies the kth support by

$$\texttt{Support\_k} = [\texttt{node\_no} \quad \texttt{support\_type} \ a] \tag{7.8}$$

where `node_no` is the node number of the support, `support_type` is an integer between 1 and 4 identifying one of the four supports (S1 to S4) shown in Fig. 7.2.3,

TABLE 7.3.1 Specification of Supports

Support		*Support_k*
(S1) Pin or hinge $U = 0,\ V = 0$		[node_no 1 0]
(S2) Roller on a horizontal surface $V = 0$		[node_no 2 0]
(S3) Roller on a vertical surface $U = 0$		[node_no 3 0]
(S4) Roller on an incline of angle α $V_n = 0$		[node_no 4 α] Angle α is given in degrees.

and a is zero for supports S1 to S3 and an incline angle for support S4. Table 7.3.1 lists the entries of `Support_k` for these types of supports.

Note. The information of a truss set up by function `settruss` can be accessed at any time by typing

```
>> systinfo
```

EXAMPLE 3.1

Consider the truss in Fig. 7.3.1. Let every member have the same longitudinal rigidity EA = 800 kN. Assume that the angle of the incline at node 5 is 30 degrees. The matrices for system setup are

$$\texttt{Member\_Spec} = \begin{bmatrix} 8\cdot 10^8 & 1 & 2 \\ 8\cdot 10^8 & 2 & 4 \\ 8\cdot 10^8 & 4 & 5 \\ 8\cdot 10^8 & 3 & 5 \\ 8\cdot 10^8 & 1 & 3 \\ 8\cdot 10^8 & 2 & 3 \\ 8\cdot 10^8 & 3 & 4 \end{bmatrix}, \quad \texttt{Node\_Spec} = \begin{bmatrix} 0 & 0 \\ 1 & 1.7 \\ 0 & 2 \\ 3 & 1.7 \\ 4 & 0 \end{bmatrix},$$

$$\text{Support\_Spec} = \begin{bmatrix} 1 & 1 & 0 \\ 3 & 2 & 0 \\ 5 & 4 & 30 \end{bmatrix}.$$

The truss structure is set up by the following commands typed in the MATLAB command window:

```
>> EA = 800e3;
>> Member_Spec = [EA 1 2; EA 2 4; EA 4 5; EA 3 5; EA 1 3; EA 2 3;
                  EA 3 4];
>> Node_Spec = [0 0; 1 1.7; 2 0; 3 1.7; 4 0];
>> Support_Spec = [1 1 0; 3 2 0; 5 4 30];
>> settruss(Member_Spec, Node_Spec, Support_Spec)
```

which yield the following information and plot the undeformed configuration of the truss (not shown).

```
STATIC ANALYSIS OF A PLANE TRUSS

Description of the Truss
------------------------
Members (total number = 7)
  Member 1: EA = 800000, L = 1.9723, left node = 1, right node = 2
  Member 2: EA = 800000, L = 2, left node = 2, right node = 4
  Member 3: EA = 800000, L = 1.9723, left node = 4, right node = 5
  Member 4: EA = 800000, L = 2, left node = 3, right node = 5
  Member 5: EA = 800000, L = 2, left node = 1, right node = 3
  Member 6: EA = 800000, L = 1.9723, left node = 2, right node = 3
  Member 7: EA = 800000, L = 1.9723, left node = 3, right node = 4

  Here EA = stiffness, L = length
    left node = node number of left end of the member
    right node = node number of right end of the member

Nodes (total number = 5)
  Node 1: X = 0, Y = 0
  Node 2: X = 1, Y = 1.7
  Node 3: X = 2, Y = 0
  Node 4: X = 3, Y = 1.7
  Node 5: X = 4, Y = 0

Supports:
  at node 1, pin (hinge) with U = 0 and V = 0
  at node 3, roller placed on a horizontal surface, V = 0
  at node 5, roller placed on an inclined surface with
             angle = 30 deg.
             Vn (displacement normal to the surface) = 0
```

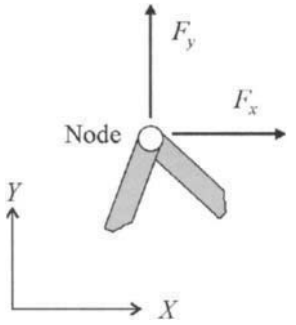


FIGURE 7.3.3 Concentrated external loads.

7.3.2 Response to External Forces

As mentioned previously, a truss carries concentrated external loads only at its nodes. Figure 7.3.3 shows the sign convention of external loads at a node, where positive forces F_x and F_y are in the directions of the positive X and Y axes, respectively. The Toolbox of this chapter provides a function `trussresp` for computing the response of a truss subject to concentrated loads; see Windows 3.2.

Window 3.2. Function trussresp for Response to External Loads

MATLAB Function: **trussresp**

Purpose:
To compute and display the static response of a truss structure under external loads.

Synopsis:

```
trussresp(Load_Spec)
[U_nodal, F_axial, Rc] = trussresp(Load_Spec)
```

Description:
`trussresp(Load_Spec)` determines the response of a truss under concentrated external loads and plots its deformed configuration. The `Load_Spec` is a matrix describing external forces by

$$\texttt{Load\_Spec} = \begin{bmatrix} \texttt{Load\_1} \\ \texttt{Load\_2} \\ \vdots \\ \texttt{Load\_Nf} \end{bmatrix}$$

where `Nf` is the number of the loads. A typical row of the matrix is of the form

```
Load_j = [node_no Fx Fy]
```

where `node_no` is a node number identifying the node at which external loads are applied, and `Fx` and `Fy` are the horizontal and vertical forces as shown in Fig. 7.3.3.

Window 3.2. Function trussresp for Response to External Loads (*Continued*)

`[U_nodal, F_axial, Rc] = trussresp(Load_Spec)` returns the computed nodal displacements in a two-column matrix

$$\texttt{U\_nodal} = \begin{bmatrix} U_1 & V_1 \\ U_2 & V_2 \\ \vdots & \vdots \\ U_N & V_N \end{bmatrix}$$

where U_i and V_i are the horizontal and vertical displacements of the ith node, the axial forces of members in a vector

$$\texttt{F\_axial} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_M \end{bmatrix}$$

where $S_i = EA_i du/dx$ is the axial force of the ith member, and the reactions at the truss supports in a matrix

$$\texttt{Rc} = \begin{bmatrix} \texttt{node\_1} & R_1 \\ \texttt{node\_2} & R_2 \\ \vdots & \vdots \\ \texttt{node\_Nr} & R_r \end{bmatrix}$$

where `Nr` is the total number of the reactions, and R_j is a reaction at node `node_j`.

EXAMPLE 3.2

The truss in Example 3.1 is subject to a load at node 4, $F_x = -20$ kN and $F_y = -10$ kN; see Fig. 7.3.4. The load specification vector for function `trussresp` is

```
Load_Spec = [4 -10000 -20000]
```

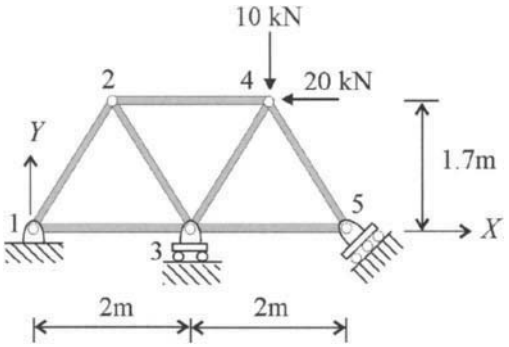


FIGURE 7.3.4

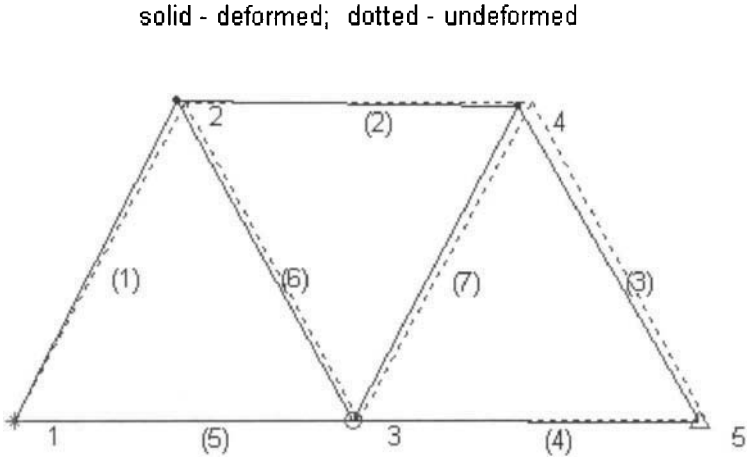


FIGURE 7.3.5

Assume that the truss has been set up by function `settruss`, as in Example 3.1. The response of the truss under the external load is computed by the commands

```
>> Load_Spec = [4 -2e4 -1e4];
>> trussresp(Load_Spec)
```

which yield the truss configurations plotted in Fig. 7.3.5 and the following results:

```
Static Response of the Truss
----------------------------

INPUT DATA:

External loads
   at node 4: Fx = -20000, Fy = -10000

COMPUTED RESULTS:

Nodal displacements U & V in global coordinates X, Y
   node 1: U = 0, V = 0
   node 2: U = -0.054404, V = 0.0095575
   node 3: U = -0.032495, V = 0
   node 4: U = -0.074297, V = -0.025972
   node 5: U = -0.032638, V = -0.018844

Displacements of rollers on inclines
   at node 5 (an inclined surface of angle = 30 deg):
      tangential displacement Ut = -0.037687, normal displacement
      Vn = 0

Internal forces of members
   Member No.   Axial Force
       1         -7847.0372
       2         -7957.2116
```

```
        3        6075.139
        4        -56.998
        5        -12998.1745
        6        7847.0372
        7        -17676.9525

Reactions
   at node 1: Rx = 16976.7803
   at node 1: Ry = 6763.6298
   at node 3: Ry = 8472.7403
   at node 5: Rn = -6046.4394, normal to an inclined surface of
              angle = 30 deg
```

7.3.3 Support Settlement

For a truss, the foundation of its supports may undergo deflections or displacements. This phenomenon is called *settlement of supports*. Under support settlement, the configuration of a truss will change. For a statically indeterminate truss, support settlement will cause elastic deformation, and thus induce internal forces in the members. Function `trussresp` from the Toolbox can be used to compute the static response of a truss under support settlement; see Window 3.4.

Window 3.3. Function `trussresp` for Response to Support Settlement

MATLAB Function: **`trussresp`**

Purpose:
To compute and display the response of a truss under settlement of supports.

Synopsis:

```
trussresp(0, SD_Spec)
[U_nodal, F_axial, Rc] = trussresp(0, SD_Spec)
```

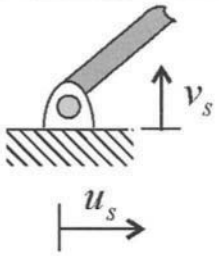
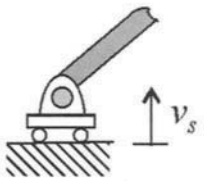
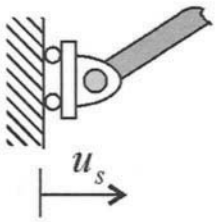
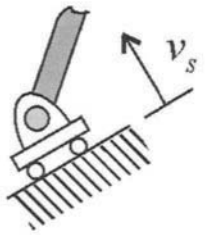
Description:
`trussresp(0, SD_Spec)` determines the response of a truss under settlement of supports. Here, `SD_Spec` is a matrix specifying support settlement by

$$\texttt{SD\_Spec} = \begin{bmatrix} \texttt{SD\_1} \\ \texttt{SD\_2} \\ \vdots \\ \texttt{SD\_Ns} \end{bmatrix}$$

where `Ns` is the number of support displacements, and each row of the matrix specifies a support displacement. A typical row of `SD_Spec` is of the form

```
SD_k = [node_no  support_type  u1 u2]
```

TABLE 7.3.2 Specification of Support Settlement

Settlement of Support		*SD_k*
(S1) Pin or hinge $U = u_s, V = v_s$		[node_no 1 u_s v_s]
(S2) Roller on a horizontal surface $V = v_s$		[node_no 2 v_s 0]
(S3) Roller on a vertical surface $U = u_s$		[node_no 3 u_s 0]
(S4) Roller on an incline of angle α $V_n = v_s$		[node_no 4 v_s 0]

Window 3.3. Function `trussresp` for Response Due to Support Settlement (*Continued*)

where `node_no` is node number, `support_type` is an integer between 1 and 4 specifying one of the four types of supports (S1 to S4 in Fig. 7.2.3), and the parameters `u1` and `u2` describe support displacements. See Table 7.3.2 for the entries of `SD_k`.

`[U_nodal, F_axial, Rc] = trussresp(0, SD_Spec)` returns the computed nodal displacements, internal forces, and reactions in matrices `U_nodal`, `F_axial`, and `Rc`, respectively, as explained in Window 3.2.

EXAMPLE 3.3

Consider the same truss as in Example 3.1. Assume that node 3 (support of Type S2) experiences a vertical displacement $v_s = -0.05$ m; see Fig. 7.3.6. The specification vector for the support settlement is

```
SD_Spec = [3 2 -0.05 0]
```

Let the structure be set up as in Example 3.1. The truss response due to the support settlement is obtained by

```
>> trussresp(0, [3 2 -0.05 0])
```

which yields the truss configurations in Fig. 7.3.7 and the following results:

```
Nodal displacements U & V in global coordinates X, Y
   node 1: U = 0, V = 0
   node 2: U = 0.0088705,  V = -0.025048
   node 3: U = 0.00016261, V = -0.05
   node 4: U = -0.0087049, V = -0.024954
   node 5: U = 0.00032522, V = 0.00018777

Displacements of rollers on inclines
   at node 5 (an inclined surface of angle = 30 deg):
   tangential displacement Ut = 0.00037554, normal displacement
   Vn = 0

Internal forces of members
   Member No.   Axial Force
      1          -6932.8135
      2          -7030.1519
      3          -6932.8135
      4          65.0449
      5          65.0449
      6          6932.8135
      7          6932.8135
```

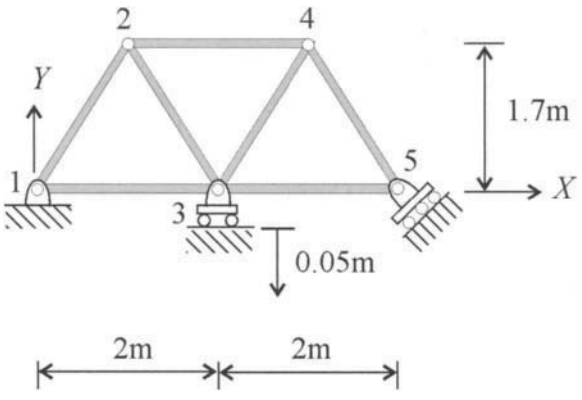


FIGURE 7.3.6 A truss with support settlement at node 3.

```
Reactions
   at node 1: Rx = 3450.0311
   at node 1: Ry = 5975.6291
   at node 3: Ry = -11951.2583
   at node 5: Rn = 6900.0622, normal to an inclined surface
                of angle = 30 deg
```

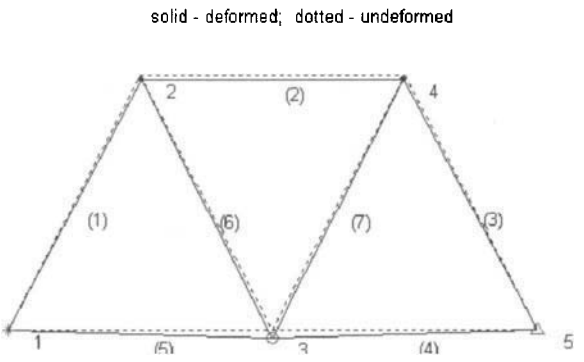


FIGURE 7.3.7

7.3.4 Fabrication Errors and Thermal Effects

Due to fabrication errors and thermal effects, some members of a truss may be subject to changes in length, which in turn can cause elastic deformation of the entire structure. If a member is fabricated too long by an amount ΔL, as shown in Fig. 7.3.8a, it will cause a compressive internal force in the member after it is fitted into a truss. On the other hand, if a member of length L is subject to a temperature increase ΔT (see Fig. 7.3.8b), it will undergo an increase in length

$$\Delta L = \alpha \Delta T L \tag{7.9}$$

where α is the coefficient of thermal expansion. This increase in length also induces a compressive internal force in the member. Similarly, decrease in length (due to either fabrication error or thermal effects) will result in tensile internal forces in members.

In this Toolbox, function `trussresp` can be called to compute the response of a truss due to the above-mentioned changes in length; see Windows 3.4 and 3.5.

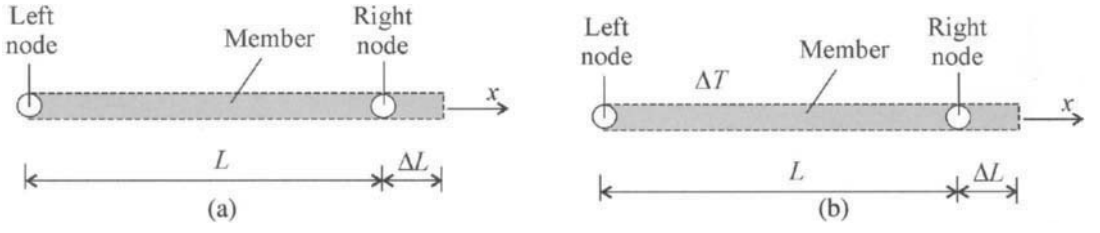


FIGURE 7.3.8 Truss member with (a) fabrication errors and (b) thermal changes.

Window 3.4. Function trussresp for Response Due to Fabrication Errors

MATLAB Function: **trussresp**

Purpose:
To compute and display the response of a truss due to fabrication errors.

Synopsis:

```
trussresp(0, 0, Fab_Errors)
[U_nodal, F_axial, Rc] = trussresp(0, 0, Fab_Errors)
```

Description:
`trussresp(0, 0, Fab_Errors)` determines the static response of a truss under fabrication errors in members and plots its deformed configuration. The `Fab_Errors` is a matrix of fabrication errors, which is given by

$$\texttt{Fab\_Errors} = \begin{bmatrix} \texttt{FE\_1} \\ \texttt{FE\_2} \\ \vdots \\ \texttt{FE\_Ne} \end{bmatrix}$$

where `Ne` is the number of members with fabrication errors. A typical row of the matrix has the form

$$\texttt{FE\_k} = [\texttt{member\_no}\ \Delta L]$$

where `member_no` is member identification number, and ΔL is amount of change in length.

`[U_nodal, F_axial, Rc] = trussresp(0, 0, Fab_Errors)` returns the computed nodal displacements, internal forces, and reactions in matrices `U_nodal`, `F_axial`, and `Rc`, respectively, as explained in Window 3.2.

Window 3.5. Function trussresp for Response to Thermal Changes

MATLAB Function: **trussresp**

Purpose:
To compute and display the response of a truss subject to thermal changes.

Synopsis:

```
trussresp(0, 0, 0, Therm_Effects)
[U_nodal, F_axial, Rc] = trussresp(0, 0, 0, Therm_Effects)
```

Window 3.5. Function trussresp for Response to Thermal Changes (*Continued*)

Description:
`trussresp(0, 0, 0, Therm_Effects)` determines the static response of a truss subject to thermal changes and plots its deformed configuration. The `Therm_Effects` is a matrix of thermal effects, which is given by

$$\texttt{Therm\_Effects} = \begin{bmatrix} \texttt{TE\_1} \\ \texttt{TE\_2} \\ \vdots \\ \texttt{TE\_Nt} \end{bmatrix}$$

where `Nt` is the number of members with fabrication errors. A typical row of the matrix has the form

$$\texttt{TE\_k} = [\texttt{member\_no}\ \alpha\ \Delta T]$$

where `member_no` is member identification number, α is coefficient of thermal expansion, and ΔT is a temperature increase.

`[U_nodal, F_axial, Rc] = trussresp(0, 0, 0, Therm_Effects)` returns the computed nodal displacements, internal forces, and reactions in matrices `U_nodal`, `F_axial`, and `Rc`, respectively, as explained in Window 3.2.

EXAMPLE 3.4

Consider the same truss structure as in Example 3.1. Let members 2 and 3 have fabrication errors as follows: member 2 is 0.05 m too short, i.e., $\Delta L = -0.05$ m; and member 3 is 0.06 m too long, i.e., $\Delta L = 0.06$ m. The matrix of fabrication errors for function `trussresp` is

$$\texttt{Fab\_Errors} = \begin{bmatrix} 2 & -0.05 \\ 3 & 0.06 \end{bmatrix}$$

Let the truss be set up by `settruss` as in Example 3.1. The response of the truss to such fabrication errors is computed by the commands

```
>> Fab_Errors = [2 -0.05; 3 0.06];
>> trussresp(0, 0, Fab_Errors)
```

which yield Fig. 7.3.9 and the results below:

```
Nodal displacements U & V in global coordinates X, Y
   node 1: U = 0, V = 0
   node 2: U = -0.0036277,  V = -5.1593e-006
   node 3: U = 1.7542e-005, V = 0
   node 4: U = -0.055524,   V = 0.03481
   node 5: U = 3.5083e-005, V = 2.0255e-005
```

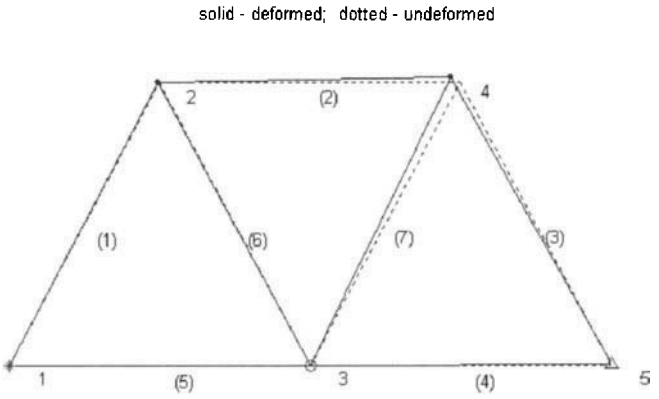


FIGURE 7.3.9

```
Displacements of rollers on inclines
   at node 5 (an inclined surface of angle = 30 deg):
      tangential displacement Ut = 4.051e-005, normal displacement
      Vn = 0

Internal forces of members
   Member No.   Axial Force
      1          -747.867
      2          -758.3672
      3          -747.867
      4          7.0166
      5          7.0166
      6          747.867
      7          747.867

Reactions
   at node 1: Rx = 372.167
   at node 1: Ry = 644.6121
   at node 3: Ry = -1289.2242
   at node 5: Rn = 744.334, normal to an inclined surface of
              angle = 30 deg
```

EXAMPLE 3.5

Consider the same truss structure as in Example 3.1. Member 6 is subject to a temperature increase $\Delta T = 150°F$. Assume the coefficient of thermal expansion of the member is $\alpha = 3.1 \times 10^{-4}/°F$. The matrix of thermal effects for function `trussresp` is

$$\texttt{Therm\_Effects} = [6 \ 3.1 \times 10^{-4} \ 150].$$

Assume that the truss has been set up by `settruss`, as in Example 3.1. The response of the truss to the temperature increase is computed by the commands

```
>> Therm_Effects = [6 3.1e-4 150];
>> trussresp(0, 0, 0, Therm_Effects)
```

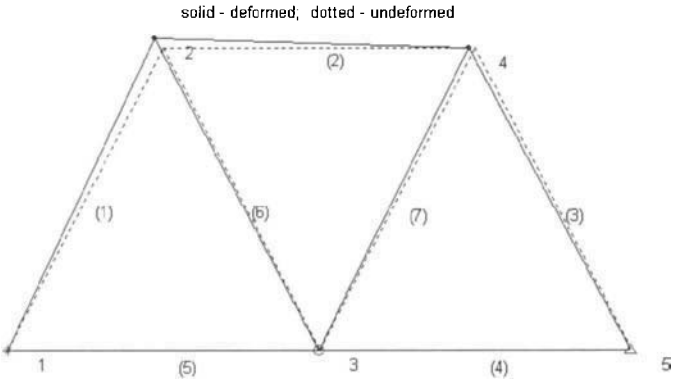


FIGURE 7.3.10

which yield Fig. 7.3.10 and the results below:

```
Nodal displacements U & V in global coordinates X, Y
   node 1: U = 0, V = 0
   node 2: U = -0.05466,     V = 0.053252
   node 3: U = -0.00017302, V = 0
   node 4: U = -0.035959,    V = -4.9006e-005
   node 5: U = -0.00034605, V = -0.00019979

Displacements of rollers on inclines
   at node 5 (an inclined surface of angle = 30 deg):
      tangential displacement Ut = -0.00039958, normal
      displacement Vn = 0

Internal forces of members
   Member No.   Axial Force
      1         7376.7174
      2         7480.2884
      3         7376.7174
      4         -69.2097
      5         -69.2097
      6         -7376.7174
      7         -7376.7174

Reactions
   at node 1: Rx = -3670.9346
   at node 1: Ry = -6358.2452
   at node 3: Ry = 12716.4903
   at node 5: Rn = -7341.8691, normal to an inclined surface of
              angle = 30 deg
```

7.3.5 Total Response

When a truss is subject to external loads, support settlement, fabrication errors, and thermal changes simultaneously, the total static response of the structure can be obtained by the principle

of superposition. Function `trussresp` from the Toolbox is capable of obtaining the total response, as shown in Window 3.6.

Window 3.6. Function trussresp for Total Response

MATLAB Function: **trussresp**

Purpose:
To compute and display the total response of a truss.

Synopsis:

```
trussresp(Load_Spec, SD_Spec, Fab_Errors, Therm_Effects)
[U_nodal, F_axial, Rc] = trussresp(Load_Spec, SD_Spec, Fab_Errors,
Therm_Effects)
```

Description:
`trussresp(Load_Spec, SD_Spec, Fab_Errors, Therm_Effects)` determines the total response of a truss subject to the combined disturbances of concentrated external loads described by `Load_Spec`, support settlement described by `SD_Spec`, fabrication errors described by `Fab_Errors`, and thermal changes described by `Therm_Effects`. The specifications of matrices `Load_Spec`, `SD_Spec`, `Fab_Errors`, `Therm_Effects` have been given in Windows 3.2 to 3.5. Any one of the matrices can be replaced by 0 or [ ] as a placeholder if the corresponding disturbance does not exist. For instance, if a truss is subject to external loads and fabrication errors in its members, its response is obtained by

```
trussresp(Load_Spec, 0, Fab_Errors) or
trussresp(Load_Spec, [ ], Fab_Errors)
```

`[U_nodal, F_axial, Rc] = trussresp(Load_Spec, SD_Spec, Fab_Errors, Therm_Effects)` returns the computed nodal displacements, internal forces, and reactions in matrices `U_nodal`, `F_axial`, and `Rc`, respectively, as explained in Window 3.2.

EXAMPLE 3.6

In Fig. 7.3.11, the same truss given in Example 3.1 is subject to two external loads at nodes 2 and 4, support settlement at node 5 ($v_s = -0.07$ m), and a fabrication error in member 1 ($\Delta L = -0.025$ m). Assume that the truss has been set up by `settruss` as in Example 3.1. The total response of the truss is computed by

```
>> load = [2 0 -3e4; 4 0 -2.5e4];
>> sd = [5 4 -0.07];
>> fab_errors = [1 -0.025];
>> trussresp(load, sd, fab_errors);
```

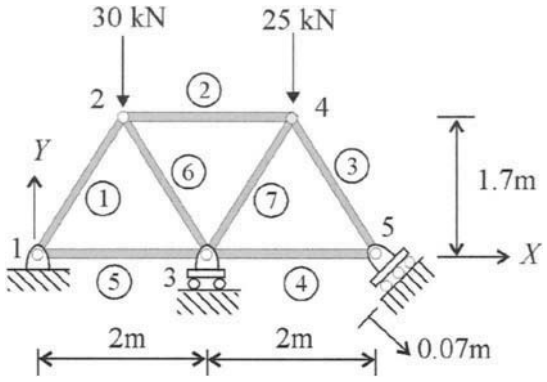


FIGURE 7.3.11

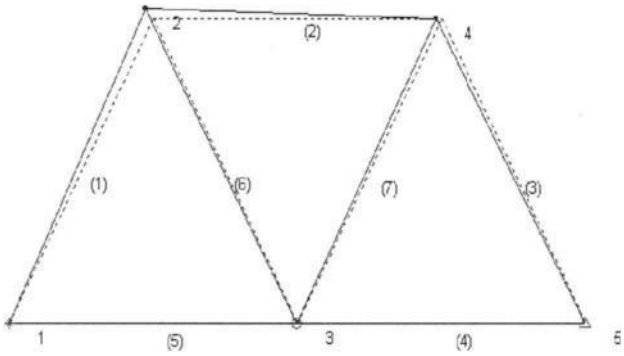


FIGURE 7.3.12

which yields Fig. 7.3.12 and the results below:

```
Nodal displacements U & V in global coordinates X, Y
   node 1: U = 0, V = 0
   node 2: U = 0.014307,  V = -0.065408
   node 3: U = 0.0038379, V = 0
   node 4: U = 0.033619,  V = -0.080788
   node 5: U = 0.0039994, V = -0.07852

Displacements of rollers on inclines
   at node 5 (an inclined surface of angle = 30 deg):
      tangential displacement Ut = -0.035796, normal
      displacement Vn = -0.07

Internal forces of members
   Member No.    Axial Force
      1          -9784.8758
      2          7724.8009
      3          -6884.4224
      4          64.5909
      5          1535.1791
```

```
        6          -25020.5647
        7          -22120.1113

   Reactions
      at node 1: Rx = 3425.9499
      at node 1: Ry = 8433.9192
      at node 3: Ry = 40632.1615
      at node 5: Rn = 6851.8997, normal to an inclined surface of
                 angle = 30 deg
```

To suppress member and node numbers in plotting the truss configurations, use the command

```
>> plottruss(2,0)
```

which produces Fig. 7.3.13. See Window 3.7 for more information on function `plottruss`.

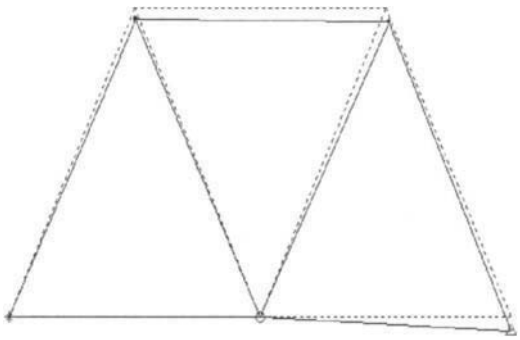


FIGURE 7.3.13

Window 3.7. Function `plottruss`

MATLAB Function: **`plottruss`**

Purpose:
To plot undeformed and deformed configurations of a truss.

Synopsis:

```
plottruss
plottruss(opt, display_ID)
```

Description:
`plottruss` plots both deformed and undeformed configurations of a truss, in which a hinge is expressed by a star (*), a roller by a circle (o), and a roller on an incline by a triangle (Δ).

Window 3.7. Function `plottruss` (*Continued*)

`plottruss(opt, display_ID)` plots truss configurations with the following options:

For parameter `opt`

`opt = 0` plot undeformed configuration only

`opt = 1` plot deformed configuration only

`opt = 2` plot both deformed and undeformed configurations

For parameter `display_ID`

`display_ID = 0` do not display member and node identification numbers

`display_ID = 1` display member and node identification numbers

Member identification numbers are given in parentheses. By default, `opt = 2` and `display_ID = 0`.

7.4 The Stiffness Method

The MATLAB Toolbox of this chapter is developed according to the stiffness method. In this method, a truss is first decomposed into a number of slender members with their ends interconnected at nodes. For each member, a stiffness matrix relating the nodal displacements to the nodal forces of the member is derived. Through coordinate transformation and force balance at nodes, a global equilibrium equation is assembled from the member stiffness matrices. Solution of the equation for nodal displacements eventually yields the static response of the truss. References 1 and 5 listed in Section 7.6 give further details on this solution method.

The above concept of the stiffness method is further explained in sequel. The user who is only interested in using the Toolbox can skip this section.

Truss Decomposition

Decomposition of a truss into members and nodes has been discussed in Section 7.3.1. The key is to specify the members and nodes by identification numbers.

Member Stiffness Matrix

Consider a typical member e with nodes i and j, as shown in Fig. 7.4.1. Define local coordinates x, y, with the positive x-axis along the longitudinal direction of the member. Let u_i and u_j be the displacements of the member at the nodes, which are called *nodal displacements*. Let q_i and q_j be the corresponding nodal forces. By the bar model given in Eq. (7.1), the nodal displacements and nodal forces are related by

$$\begin{pmatrix} q_i \\ q_j \end{pmatrix} = \left[\tilde{k}_e\right] \begin{pmatrix} u_i \\ u_j \end{pmatrix} \tag{7.10}$$

where $\left[\tilde{k}_e\right]$, the member stiffness matrix, is given by

$$\left[\tilde{k}_e\right] = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{7.11}$$

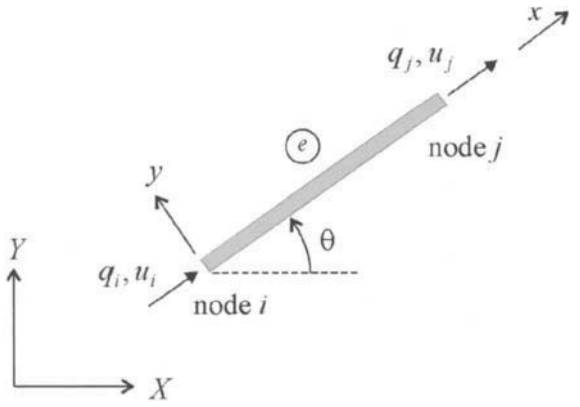


FIGURE 7.4.1 A truss member with nodal displacements and forces.

where L is the bar length. The internal (axial) force of the member is

$$S = q_j = EA\frac{u_j - u_i}{L} \tag{7.12}$$

Coordinate Transformation

To analyze nodal displacements and forces at all nodes, a global coordinate system XY is established, as shown in Fig. 7.4.1. Thus, a transformation of nodal displacements and forces from local coordinates to global coordinates is necessary. From Fig. 7.4.2, nodal displacement u_i in the local coordinates can be transformed to components U_i and V_i in the global coordinates by

$$u_i = U_i \cos\theta + V_i \sin\theta \tag{7.13}$$

Likewise, the nodal force q_i is related to the global components $Q_{X,i}$ and $Q_{Y,i}$ by

$$Q_{X,i} = q_i \cos\theta, \quad Q_{Y,i} = q_i \sin\theta \tag{7.14}$$

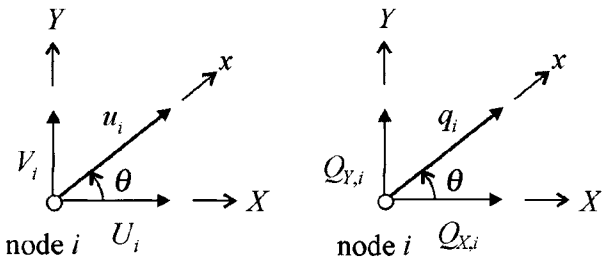


FIGURE 7.4.2 Nodal displacements and forces in global coordinates.

It follows that nodal forces and displacements in global coordinates are related by

$$\begin{pmatrix} Q_{X,i} \\ Q_{Y,i} \\ Q_{X,j} \\ Q_{Y,j} \end{pmatrix} = [k_e] \begin{pmatrix} U_i \\ V_i \\ U_j \\ V_j \end{pmatrix} \tag{7.15}$$

where $[k_e]$, the member stiffness matrix in global coordinates, is

$$[k_e] = [T]^T \left[\tilde{k}_e\right] [T] \tag{7.16}$$

and $[T]$, a coordinate transformation matrix, is

$$[T] = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \end{bmatrix} \tag{7.17}$$

The cosine and sine in the above equations are given by

$$\cos\theta = \frac{X_j - X_i}{L}, \quad \sin\theta = \frac{Y_j - Y_i}{L} \qquad \text{with } L = \sqrt{(X_j - X_i)^2 + (Y_j - Y_i)^2} \tag{7.18}$$

where (X_i, Y_i) and (X_j, Y_j) are the coordinates of nodes i and j, respectively, and L is the length of the member. Also, by Eqs. (7.12) and (7.13), the internal force of the member is

$$S = \frac{EA}{L}\left[(U_j - U_i)\cos\theta + (V_j - V_i)\sin\theta\right] \tag{7.19}$$

Assembly of Truss

The truss is assembled from its members through force balance at all nodes. Figure 7.4.3 shows a typical case of force balance at a node, where members A, B, and C are pin-connected at node i. Let the other ends of the members (not shown) be nodes j, l, and m, respectively.

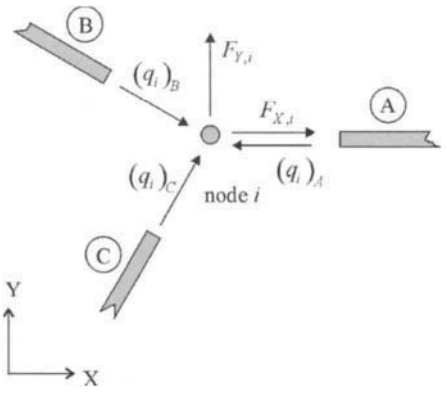


FIGURE 7.4.3 Assembly of truss members at a node.

In the figure $(q_i)_A$, $(q_i)_B$, and $(q_i)_C$ are the nodal forces applied by the members at node i, and $F_{X,i}$, $F_{Y,i}$ are concentrated external loads. The equilibrium equations at the node are

$$\begin{aligned} F_{X,i} - Q_{X,i}^{(A)} - Q_{X,i}^{(B)} - Q_{X,i}^{(C)} &= 0 \\ F_{Y,i} - Q_{Y,i}^{(A)} - Q_{Y,i}^{(B)} - Q_{Y,i}^{(C)} &= 0 \end{aligned} \tag{7.20}$$

where $Q_{X,i}^{(A)}$ and $Q_{Y,i}^{(A)}$ are the components (projections) of the nodal force $(q_i)_A$ along the global X, Y axes, as obtained from Eq. (7.14). According to Eq. (7.15), all nodal forces can be expressed in terms of nodal displacements in global coordinates. It follows that Eq. (7.20) is reduced to

$$\begin{aligned} &\left[k_A^{(i)}\right]\begin{pmatrix} U_i \\ V_i \end{pmatrix} + \left[k_B^{(i)}\right]\begin{pmatrix} U_i \\ V_i \end{pmatrix} + \left[k_C^{(i)}\right]\begin{pmatrix} U_i \\ V_i \end{pmatrix} \\ &\qquad + \left[k_A^{(j)}\right]\begin{pmatrix} U_j \\ V_j \end{pmatrix} + \left[k_B^{(l)}\right]\begin{pmatrix} U_l \\ V_l \end{pmatrix} + \left[k_C^{(m)}\right]\begin{pmatrix} U_m \\ V_m \end{pmatrix} = \begin{pmatrix} F_{X,i} \\ F_{Y,i} \end{pmatrix} \end{aligned} \tag{7.21}$$

where U_i, V_i are nodal displacements (in global coordinates) at node i, and $\left[k_A^{(i)}\right], \ldots, \left[k_C^{(m)}\right]$ are submatrices obtained from partition of the member stiffness matrices given by Eq. (7.16).

If the truss has N nodes, force balance at all the nodes leads to a global equilibrium equation

$$[K]\{U\} = \{F\} \tag{7.22}$$

where $\{U\}$ is a vector of nodal displacements, $\{U\} = \begin{pmatrix} U_1 & V_1 & \cdots & U_N & V_N \end{pmatrix}^T$; $\{F\}$ is a vector of external loads and support reactions; and $[K]$ is the global stiffness matrix of the truss that is composed of those submatrices like $\left[k_A^{(i)}\right], \ldots, \left[k_C^{(m)}\right]$ in Eq. (7.21).

Determination of Truss Response

Before Eq. (7.22) can be solved, the conditions at the supports of the truss must be introduced to eliminate known nodal displacements. If node i is a hinge (S1 in Fig. 7.2.3), $U_i = 0, V_i = 0$; and if node j is a roller on a horizontal surface which has support displacement (S2 in Table 7.3.2), $V_j = v_s$ where v_s is a known support displacement. Elimination of those known displacements at the supports from $\{U\}$, the corresponding entries from $\{F\}$, and the corresponding rows and columns from $[K]$, yields a modified equilibrium equation

$$\left[\hat{K}\right]\left\{\hat{U}\right\} = \left\{\hat{F}\right\} \tag{7.23}$$

where $\left\{\hat{U}\right\}$ only contains independent nodal displacements. Solution of Eq. (7.23) gives

$$\left\{\hat{U}\right\} = \left[\hat{K}\right]^{-1}\left\{\hat{F}\right\} \tag{7.24}$$

Substitute the determined $\left\{\hat{U}\right\}$ and the known displacements at the supports into Eqs. (7.19) and (7.22), to obtain the internal forces of truss members and the reactions.

EXAMPLE 4.1

The truss in Fig. 7.4.4 is subject to a horizontal force F at node 4. The positive x axis for each member is marked in the figure. Let all the members have the same stiffness EA. The member stiffness matrix in local coordinates and coordinate transformation matrix for each member are as follows:

Member 1:

$$\left[\tilde{k}_1\right] = \frac{EA}{\sqrt{2}L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, [T] = \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Member 2:

$$\left[\tilde{k}_2\right] = \frac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, [T] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Member 3:

$$\left[\tilde{k}_3\right] = \frac{EA}{\sqrt{2}L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, [T] = \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

The member stiffness matrices in global coordinates, by Eq. (7.16), are obtained as

$$[k_1] = \alpha\begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}, \quad [k_2] = \beta\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix},$$

$$[k_3] = \alpha\begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

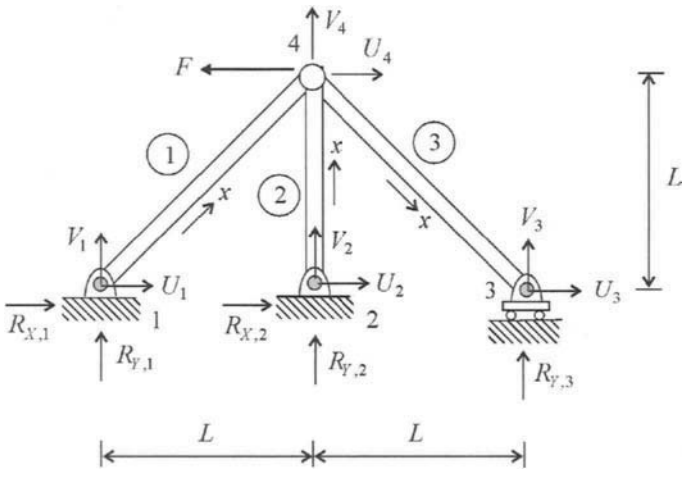


FIGURE 7.4.4

where $\alpha = \sqrt{2}\,EA/L$, $\beta = EA/L$. Assembly of the members at nodes 1 to 4 by force balance gives the global equilibrium equation of the truss

$$\begin{bmatrix} \alpha & \alpha & 0 & 0 & 0 & 0 & -\alpha & -\alpha \\ \alpha & \alpha & 0 & 0 & 0 & 0 & -\alpha & -\alpha \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \beta & 0 & 0 & 0 & -\beta \\ 0 & 0 & 0 & 0 & \alpha & -\alpha & -\alpha & \alpha \\ 0 & 0 & 0 & 0 & -\alpha & \alpha & \alpha & -\alpha \\ -\alpha & -\alpha & 0 & 0 & -\alpha & \alpha & 2\alpha & 0 \\ -\alpha & -\alpha & 0 & -\beta & \alpha & -\alpha & 0 & 2\alpha+\beta \end{bmatrix} \begin{pmatrix} U_1 \\ V_1 \\ U_2 \\ V_2 \\ U_3 \\ V_3 \\ U_4 \\ V_4 \end{pmatrix} = \begin{pmatrix} R_{X,1} \\ R_{Y,1} \\ R_{X,2} \\ R_{Y,2} \\ 0 \\ R_{Y,3} \\ -F \\ 0 \end{pmatrix}$$

where $R_{X,1}, R_{Y,1}, \ldots, R_{Y,3}$ are the reactions shown in Fig.7.4.4. The zero nodal displacements at the supports are

at node 1 (hinge): $U_1 = 0, V_1 = 0$

at node 2 (hinge): $U_2 = 0, V_2 = 0$

at node 3 (roller): $V_3 = 0$

Introduction of these known displacements to the global equilibrium equation leads to a modified equilibrium equation

$$\begin{bmatrix} \alpha & -\alpha & \alpha \\ -\alpha & 2\alpha & 0 \\ \alpha & 0 & 2\alpha+\beta \end{bmatrix} \begin{pmatrix} U_3 \\ U_4 \\ V_4 \end{pmatrix} = -F \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Note that the reactions in the unmodified equilibrium equation do not show up in the modified equilibrium equation. Solution of the above equation gives

$$U_3 = -\left(\frac{1}{\alpha} + \frac{2}{\beta}\right) F = -\left(\frac{\sqrt{2}}{2} + 2\right) \frac{FL}{EA}$$

$$U_4 = -\left(\frac{1}{\alpha} + \frac{1}{\beta}\right) F = -\left(\frac{\sqrt{2}}{2} + 1\right) \frac{FL}{EA}$$

$$V_4 = \frac{1}{\beta} F = \frac{FL}{EA}$$

Also, by Eq. (7.19), the internal forces of the members are

Member 1: $S_1 = \dfrac{EA}{2L}(U_4 + V_4) = \sqrt{2}\,F$

Member 2: $S_2 = \dfrac{EA}{L} V_4 = F$

Member 3: $S_3 = \dfrac{EA}{2L}(U_3 - U_4 + V_4) = 0$

The reactions are obtained from the unmodified global equilibrium equation:

$R_{X,1} = -\alpha(U_4 + V_4) = F$

$$R_{Y,1} = -\alpha(U_4 + V_4) = F$$

$$R_{X,2} = 0$$

$$R_{Y,2} = -\beta V_4 = -F$$

$$R_{Y,3} = -\alpha(U_3 - U_4 + V_4) = 0$$

7.5 Quick Solution Guide

System Description

A truss is a structure composed of slender members (see Fig. 7.5.1), which are pin-connected at nodes and undergo axial deformation only. External loads are applied to a truss only at nodes.

The longitudinal displacement $u(x)$ of a truss member shown in Fig. 7.5.2 is governed by

$$EA\frac{d^2}{dx^2}u(x) = 0, \quad 0 \le x \le L$$

where EA and L are the stiffness (longitudinal rigidity) and length of the member, respectively. The displacement of the member is linearly distributed:

$$u(x) = a_0 + a_1 x.$$

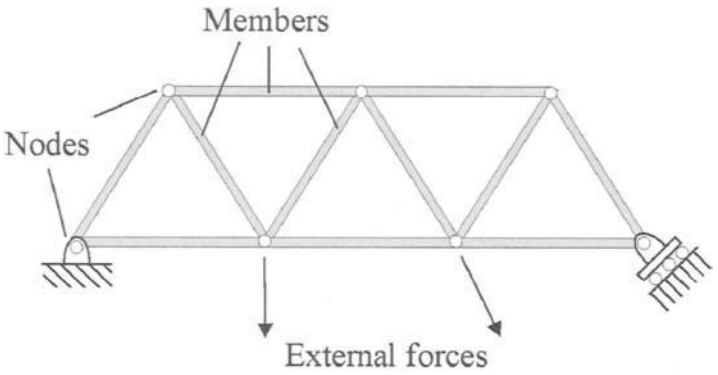


FIGURE 7.5.1 Schematic of a truss.

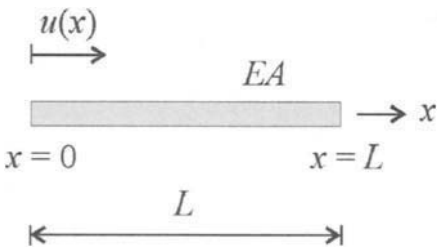


FIGURE 7.5.2 Truss member.

Because of this, the internal (axial) force of the member, $S = EAdu/dx$, is constant throughout the member.

Problems to Solve

This Toolbox has MATLAB functions for determining the solutions of the following problems of trusses:

(a) Static response to external loads;
(b) Static response to support settlement;
(c) Static response to fabrication errors in truss members; and
(d) Static response to changes of member length caused by thermal effects.

MATLAB Functions

This chapter contains a toolbox of MATLAB functions for static analysis of plane trusses; see the table below. Refer to the corresponding window or section for the utility of each function. The application of some major functions is summarized in the following pages.

Static Analysis of Trusses		
`settruss`	Set up parameters and boundary conditions of a truss	Window 3.1
`trussresp`	Compute truss response subject to external loads	Window 3.2
	Compute truss response due to support settlement	Window 3.3
	Compute truss response due to fabrication errors	Window 3.4
	Compute truss response due to thermal effects	Window 3.5
	Compute the total response of a truss	Window 3.6
`plottruss`	Display the undeformed and deformed configurations of a truss	Window 3.7
Utilities		
`systinfo`	Display truss parameters and boundary conditions	Section 7.3
`TBdemo`	Show how the Toolbox works and what it can do	Section 7.1
`TBinfo`	Show the information of the Toolbox	Section 7.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 7.1

Solution Procedure

Static analysis of a truss by the Toolbox takes the following three steps:

Step 1. Set up truss parameters and boundary conditions

```
>> settruss(Member_Spec, Node_Spec, Support_Spec)
```

Step 2. Compute static response

To compute response subject to external forces, type

```
>> trussresp(Load_Spec)
```

To compute response due to settlement of supports, type

```
>> trussresp(0, SD_Spec)
```

To compute response due to fabrication errors in truss members, type

```
>> trussresp(0, 0, Fab_Errors)
```

To compute response due to thermal changes in truss members, type

```
>> trussresp(0, 0, 0, Therm_Effects)
```

For total response, type

```
>> trussresp(Load_Spec, SD_Spec, Fab_Errors,
Therm_Effects)
```

Step 3. Plot truss configurations

```
>> plottruss
```

Note. Matrices `Member_Spec`, `Node_Spec`, `Support_Spec` specify the parameters, nodes, and supports of the truss, respectively. Matrix `Load_Spec` specifies external loads; `SD_Spec` specifies settlement of supports; `Fab_Errors` specifies fabrication errors; and `Therm_Effects` specifies thermal changes. The formation of these matrices is given below.

Specification of Truss Members

$$\texttt{Member\_Spec} = \begin{bmatrix} EA_1 & n_{L1} & n_{R1} \\ EA_2 & n_{L2} & n_{L2} \\ \vdots & \vdots & \vdots \\ EA_M & n_{LM} & n_{LM} \end{bmatrix}$$

where in the ith row, EA_i is the longitudinal rigidity of the ith member, and n_{Li} and n_{Ri} are the node numbers of the left and right ends of the member.

Specification of Nodes

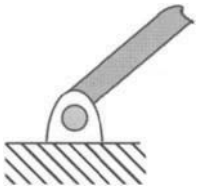
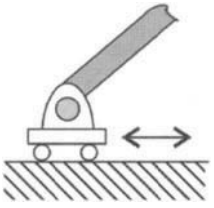
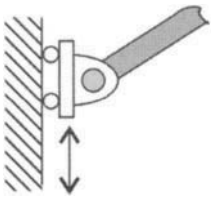
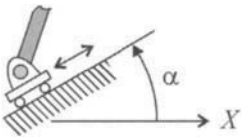
$$\texttt{Node\_Spec} = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ \vdots & \vdots \\ X_N & Y_N \end{bmatrix}$$

where the ith row of the matrix contains the global coordinates of the ith node.

Specification of Supports

$$\texttt{Support\_Spec} = \begin{bmatrix} \texttt{Support\_1} \\ \texttt{Support\_2} \\ \vdots \\ \texttt{Support\_Ns} \end{bmatrix}$$

The kth row `Support_k` of the matrix specifies the kth support according to the following table, which is the same as Table 7.3.1. The parameter `node_no` in the table is a node number.

Support	*Support_k*
(S1) Pin or hinge $U = 0, V = 0$ 	[node_no 1 0]
(S2) Roller on a horizontal surface $V = 0$ 	[node_no 2 0]
(S3) Roller on a vertical surface $U = 0$ 	[node_no 3 0]
(S4) Roller on an incline of angle α $V_n = 0$ 	[node_no 4 α] Angle α is given in degrees.

Specification of External Loads

$$\texttt{Load\_Spec} = \begin{bmatrix} \texttt{Load\_1} \\ \texttt{Load\_2} \\ \vdots \\ \texttt{Load\_Nf} \end{bmatrix}$$

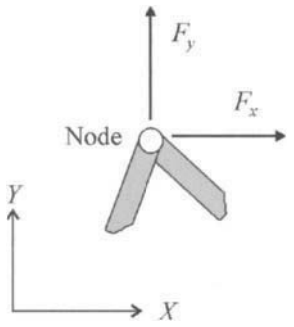


FIGURE 7.5.3 Concentrated loads at a node.

A typical row of the matrix is

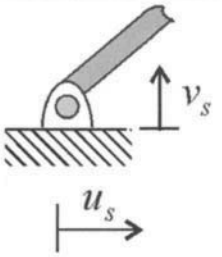
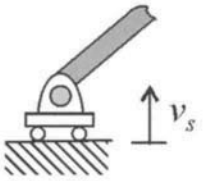
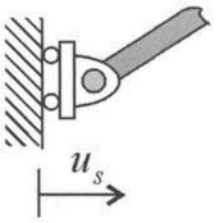
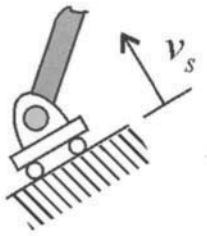
```
Load_k = [node_no Fx Fy]
```

where `node_no` is node number, `Fx` and `Fy` are the horizontal and vertical components of an external load at the node (see Fig. 7.5.3).

Specification of Displacements at Supports (settlement of supports)

$$\texttt{SD\_Spec} = \begin{bmatrix} \texttt{SD\_1} \\ \texttt{SD\_2} \\ \vdots \\ \texttt{SD\_Ns} \end{bmatrix}$$

where each row specifies a support displacement according to the following table, which is the same as Table 7.3.2. The parameter `node_no` in the table is a node number.

Settlement of Support	*SD_k*
(S1) Pin or hinge $U = u_s, V = v_s$ 	[node_no 1 u_s v_s]
(S2) Roller on a horizontal surface $V = v_s$ 	[node_no 2 v_s 0]
(S3) Roller on a vertical surface $U = u_s$ 	[node_no 3 u_s 0]
(S4) Roller on an incline of angle α $V_n = v_s$ 	[node_no 4 v_s 0]

Specification of Fabrication Errors in Truss Members

$$\texttt{Fab\_Errors} = \begin{bmatrix} \texttt{FE\_1} \\ \texttt{FE\_2} \\ \vdots \\ \texttt{FE\_Ne} \end{bmatrix}$$

where each row specifies a fabrication error in a member. A typical row of the matrix has the form

$$\texttt{FE\_k} = [\texttt{member\_no}\ \Delta L]$$

where `member_no` is member number, and ΔL is amount of change in length in the member.

Specification of Thermal Changes in Truss Members

$$\texttt{Therm\_Effects} = \begin{bmatrix} \texttt{TE\_1} \\ \texttt{TE\_2} \\ \vdots \\ \texttt{TE\_Nt} \end{bmatrix}$$

where each row specifies a thermal change in a member. A typical row of the matrix has the form

$$\texttt{TE\_k} = [\texttt{member\_no}\ \alpha\ \Delta L]$$

where `member_no` is member number, α is coefficient of thermal expansion, and ΔT is a temperature increase in the member.

7.6 References

1. Hibbeler R C 2001 *Structural Analysis*, 5th edition, Prentice-Hall: Upper Saddle River, New Jersey.
2. Kassimali A 1999 *Matrix Analysis of Structures*, Brooks Cole.
3. Przemienieck J S 1985 *Theory of Matrix Structural Analysis*, Dover Publications.
4. Mikhelson I 2004 *Structural Engineering Formulas*, McGraw-Hill: New York.
5. West H H, Geschwindner L F 2002 *Fundamentals of Structural Analysis*, 2nd edition, Wiley Text Books.

This Page Intentionally Left Blank

8 Static Analysis of Plane Frames

Inside
- Getting Started
- System Description
- Solution by the Toolbox
- The Distributed Transfer Function Method
- Quick Solution Guide
- References

8.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for static analysis of plane frames.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

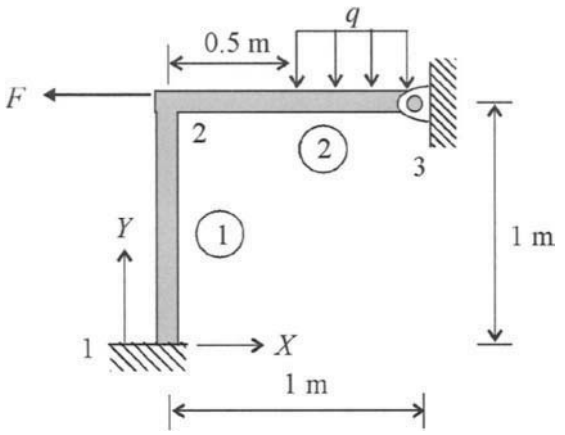


FIGURE 8.1.1 A frame under a concentrated load at node 2 and a uniform load on member 2.

Quick Tutorial

To show how to use the Toolbox, consider a frame in Fig. 8.1.1, where member identification numbers (in circles) and node identification numbers are assigned. The frame is subject to a horizontal external load ($F = 75$ kN) at node 2, and a uniform load distributed on the right-half span of member 2 ($q = 40$ kN/m). The frame is fixed at node 1 and pinned at node 3. Assume that all the members have the same stiffness parameters: $EA = 500$ kN, $EI = 700$ kN-m^2. To compute the static response of the frame, type the following in the MATLAB command window:

```
>>> EA = 500; EI = 700;
>>> members = [EA EI 1 2; EA EI 2 3];
>>> nodes = [0 0; 0 1; 1 1];
>>> supports = [1 1 0; 3 2 0];
>>> setframe(members, nodes, supports)
>>> load_interior = [2 1 0.5 1 -40];
>>> load_nodal = [2 -75 0 0];
>>> frameresp(load_interior, load_nodal)
```

where >> is the prompt in the MATLAB command window. This yields the frame configurations (undeformed and deformed) in Fig. 8.1.2 and the following results:

```
Nodal displacements U, V, and theta
 at node 1: U = 0, V = 0, theta = 0 deg
 at node 2: U = -0.023494, V = -0.028553, theta = 1.8294 deg
 at node 3: U = 0, V = 0, theta = 1.5861 deg
Reactions
 at node 1: Rx = 63.2528
 at node 1: Ry = 14.2764
 at node 1: Mc = -53.9764
 at node 3: Rx = 11.7472
 at node 3: Ry = 5.7236
```

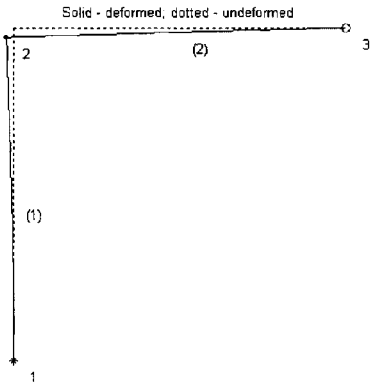


FIGURE 8.1.2 Frame configurations.

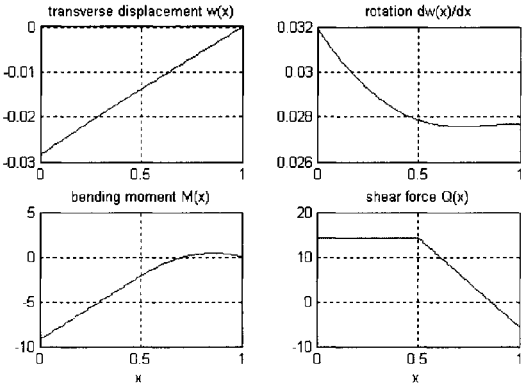


FIGURE 8.1.3 The transverse response of member 2.

To obtain the spatial distributions of the transverse response of member 2, type

```
>> plotmember(2, 4)
```

which yields Fig. 8.1.3.

In the above frame analysis, three functions from the Toolbox are called:

(a) Function **setframe** that inputs the frame parameters and assigns the supports.
(b) Function **frameresp** that computes the response of the frame subject to the external loads and displays the frame configurations; and
(c) Function **plotmember** that plots the response of a selected frame member.

These functions, along with others from the Toolbox, will be introduced in the subsequent sections.

For a quick solution, the reader may refer directly to the Quick Solution Guide (Section 8.5), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To better understand how the toolbox works, the reader is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 8.6.

8.2 System Description

A frame structure is an assembly of prismatic members that are rigidly connected at their ends; see Fig. 8.2.1 for an example. The joints where members are interconnected and the points at which the frame is supported are called *nodes*. External loads can be applied to either the nodes or the interior points of the members. Thus, a frame member may undergo both longitudinal and transverse deflections. In this section, a commonly used plane frame model is introduced (References 1 to 4).

Member Model

A frame member is modeled as an elastic prismatic element that can sustain both transverse and longitudinal deformations. The longitudinal displacement $u(x)$ and transverse displacement $w(x)$ of such a member, as shown in Fig. 8.2.2, are governed by the differential equations

$$-EA\frac{d^2}{dx^2}u(x) = f_x(x), \quad 0 \le x \le L \tag{8.1}$$

$$EI\frac{d^4}{dx^4}w(x) = f_y(x), \quad 0 \le x \le L \tag{8.2}$$

where EA and EI are the longitudinal stiffness and bending stiffness of the member, respectively; L is the member length; $f_x(x)$ is an axial external load, which causes the member to

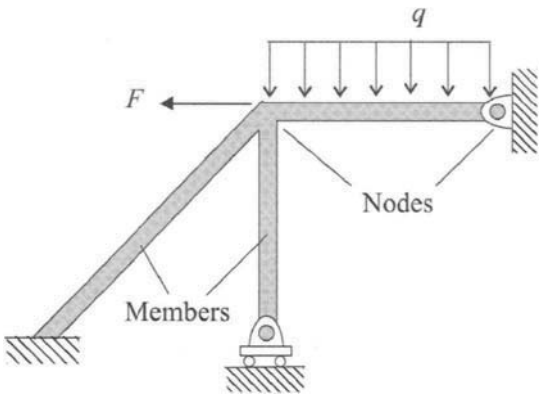


FIGURE 8.2.1 Schematic of a frame.

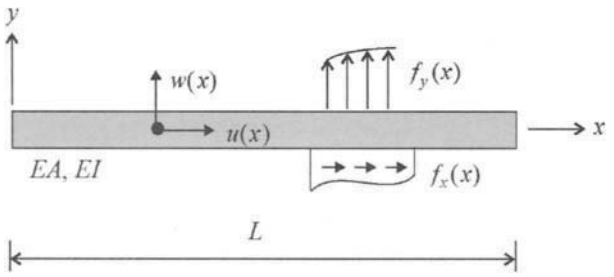


FIGURE 8.2.2 A frame member.

deform longitudinally; and $f_y(x)$ is a transverse external load, which causes the member to deform transversely. The internal forces of the member include

$$\text{axial force} \qquad S(x) = EA\frac{d}{dx}u(x) \tag{8.3a}$$

$$\text{bending moment} \qquad M(x) = EI\frac{d^2}{dx^2}w(x) \tag{8.3b}$$

$$\text{shear force} \qquad Q(x) = EI\frac{d^3}{dx^3}w(x) \tag{8.3c}$$

Figure 8.2.3 shows the sign convention of positive internal forces, which is commonly adopted in mechanics of materials.

Forces at the Ends of a Member

As mentioned previously, a frame member is rigidly connected to two nodes (joints), say nodes i and j; see Fig. 8.2.4a. The nodes impose displacements and rotations $(u_i, w_i, \theta_i, u_j, w_j, \theta_j)$ at the two ends of the member, where the rotations are given by $\theta_i = dw(0)/dx$ and $\theta_j = dw(L)/dx$. These displacements are called *nodal displacements*. Due to these nodal displacements, forces $q_{xi}, q_{yi}, q_{xj}, q_{yj}$ and moments q_{mi}, q_{mj} are developed at the ends of the member; see Fig. 8.2.4b. Here the sign convention for the nodal forces in Fig. 8.2.4b is different from that for the

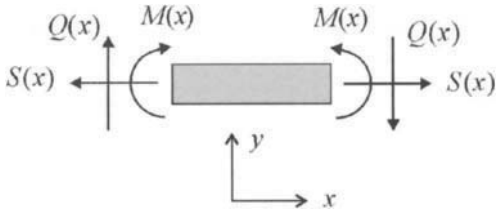


FIGURE 8.2.3 Sign convention of internal forces.

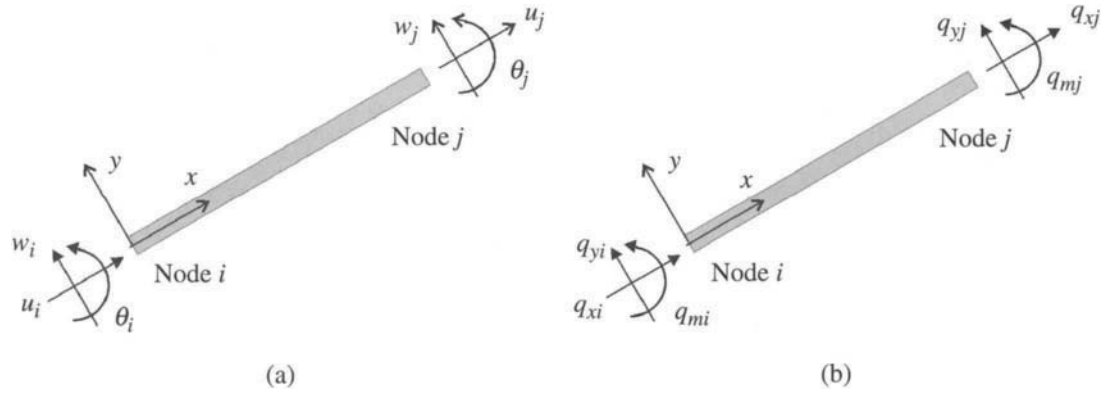


FIGURE 8.2.4 Nodal displacements and nodal forces.

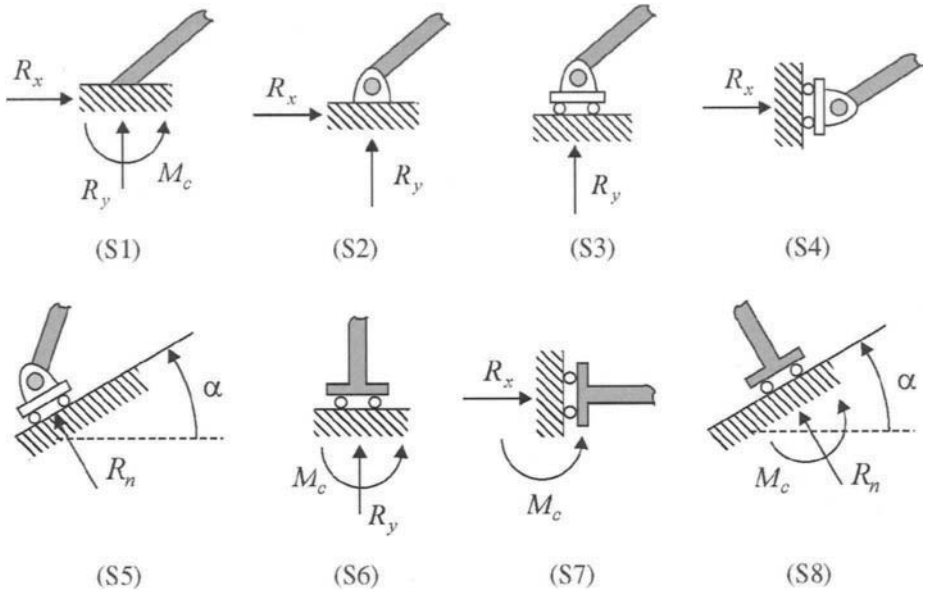


FIGURE 8.2.5 Eight types of supports.

internal forces in Fig. 8.2.3; namely,

$$
\begin{aligned}
p_{xi} &= -S(0) = -EA\frac{d}{dx}u(0), \quad p_{xj} = S(L) = EA\frac{d}{dx}u(L) \\
p_{yi} &= Q(0) = EI\frac{d^3}{dx^3}w(0), \quad p_{yj} = -Q(L) = -EI\frac{d^3}{dx^3}w(L) \\
p_{mi} &= -M(0) = -EI\frac{d^2}{dx^2}w(0), \quad p_{mj} = M(L) = EI\frac{d^2}{dx^2}w(L)
\end{aligned}
\tag{8.4}
$$

Supports

Eight types of supports for frames are shown in Fig. 8.2.5, where R_x, R_y, R_n, and M_c are the reaction forces and moments. Denote the horizontal and vertical displacements of a support by U and V, and the rotation by θ. The description of the supports is given as follows:

(S1) *Fixed end.* The horizontal and vertical displacements U, V, and rotation θ of a frame member at the support are all zero.

(S2) *Pin or hinge.* The displacements U and V of a member at the support are both zero. The member, however, can rotate at the hinge.

(S3) *Roller on a horizontal surface.* The vertical displacement V of a member at the support is zero. The roller can move horizontally, and the member can rotate at the support.

(S4) *Roller on a vertical surface.* The horizontal displacement U of a member at the support is zero. The roller can move vertically, and the member can rotate at the support.

(S5) *Roller on an incline of angle* α. The displacement V_n of the member in the normal direction of the incline surface is zero. The roller can move tangentially along the incline surface, and the member can rotate at the support.

(S6) *Slider on a horizontal surface.* The vertical displacement V and rotation θ of a member at the support are both zero. The roller can move horizontally.
(S7) *Slider on a vertical surface.* The horizontal displacement U and rotation θ of a member at the support are both zero. The roller can move vertically.
(S8) *Slider on an incline of angle* α. The displacement V_n in the normal direction of the incline surface and the rotation θ of the member are both zero. The roller can move tangentially along the incline surface.

Static Disturbances
In this chapter, the following three types of disturbances are considered:

- Concentrated loads applied to the nodes (such as F in Fig. 8.2.1);
- External forces applied to the interior points of frame members (such as q in Fig. 8.2.1);
- Settlement of supports (deflections of supports in the directions of reactions).

A detailed description of these disturbances is given in Section 8.3.

Fundamental Problem and Solution
A fundamental problem of a plane frame is to determine its static response (displacements, internal forces, and reactions at supports) subject to external loads and support settlement. This chapter provides a Toolbox of MATLAB functions for solution of the problem; see Section 8.3. The Toolbox is developed according to an exact analytical solution method; see Section 8.4.

8.3 Solution by the Toolbox

Static analysis of a frame structure via the Toolbox takes the following three steps:

Step 1. Set up the frame by function **`setframe`**;
Step 2. Determine the frame response by function **`frameresp`**; and
Step 3. Plot the response of each member by function **`plotmember`**.

These functions, along with others from the Toolbox, will be presented subsequently.

8.3.1 System Setup

To describe a frame structure, the members and nodes need to be identified by numbers. As an example, consider the frame shown in Fig. 8.3.1, which has three members identified by those numbers in circles, and four nodes marked by numbers 1 to 4. In general, for a frame of M members and N nodes, M member identification numbers and N node identification numbers are assigned in an increasing order, starting from one.

Also, the location of a node is described by a pair of global coordinates X and Y. For the frame in Fig. 8.3.1, the coordinates of the nodes are as follows:

$$\text{node 1 } (X = 0,\ Y = 0),\ \text{node 2 } (X = a,\ Y = 0)$$
$$\text{node 3 } (X = a,\ Y = a),\ \text{node 4 } (X = 2a,\ Y = a).$$

With the node coordinates assigned, the location, length, and orientation of each member are also determined.

With the member and node identification numbers and node coordinates, function `setframe` from the Toolbox is called to set up the structure; see Window 3.1.

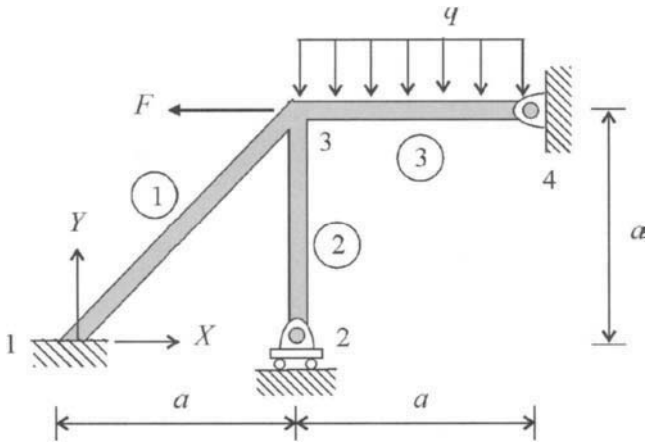


FIGURE 8.3.1 Schematic of a frame structure.

Window 3.1. Function `setframe`

MATLAB Function: **`setframe`**

Purpose:
To set up a frame structure.

Synopsis:

```
setframe(Member_Spec, Node_Spec, Support_Spec)
setframe(Member_Spec, Node_Spec, Support_Spec, npts)
```

Description:
`setframe(Member_Spec, Node_Spec, Support_Spec)` undertakes the following tasks:

(a) It inputs the stiffness parameters and node numbers of the frame members by a matrix `Member_Spec`;
(b) It inputs the coordinates of all nodes by a matrix `Node_Spec`;
(c) It specifies the supports of the frame by a matrix `Support_Spec`; and
(d) It plots the undeformed configuration of the frame.

`setframe(Member_Spec, Node_Spec, Support_Spec, npts)`, besides undertaking the above tasks, assigns the number `npts` of equally-spaced points along each frame member for simulation and graphic purposes. By default, `npts` = 101.

Note: This function must be called before other functions can be used.

In Window 3.1, function `setframe` specifies a frame structure by the following matrices:

(i) The frame members are specified by a matrix

$$\texttt{Member\_Spec} = \begin{bmatrix} EA_1 & EI_1 & n_{L1} & n_{R1} \\ EA_2 & EI_2 & n_{L2} & n_{L2} \\ \vdots & \vdots & \vdots & \vdots \\ EA_M & EI_M & n_{LM} & n_{LM} \end{bmatrix} \tag{8.5}$$

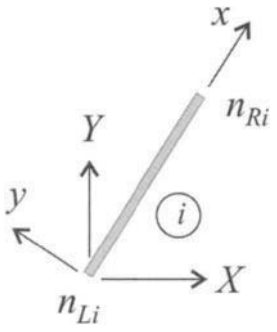


FIGURE 8.3.2 A member with local and global coordinates.

where in the ith row, EA_i and EI_i are the longitudinal rigidity and bending stiffness of member i, and n_{Li} and n_{Ri} are the node numbers of the left and right ends of the member, respectively. The node numbers also define a local coordinate system (x, y), with the positive x-axis oriented from node n_{Li} to node n_{Ri}; see Fig. 8.3.2.

(ii) The global coordinates of all the nodes are specified by a matrix

$$\texttt{Node\_Spec} = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ \vdots & \vdots \\ X_N & Y_N \end{bmatrix} \tag{8.6}$$

where the ith row of the matrix contains the global coordinates of the ith node.

(iii) The supports of the frame are specified by a matrix

$$\texttt{Support\_Spec} = \begin{bmatrix} \texttt{Support\_1} \\ \texttt{Support\_2} \\ \vdots \\ \texttt{Support\_Ns} \end{bmatrix} \tag{8.7}$$

where `Ns` is the total number of supports. The kth row of the matrix specifies the kth support by

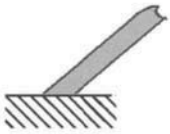
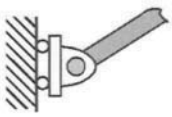
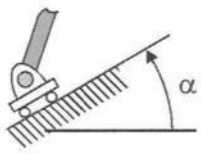
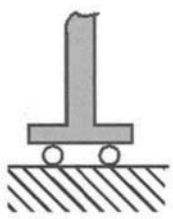
$$\texttt{Support\_k} = [\texttt{node\_no support\_type } a] \tag{8.8}$$

where `node_no` is the node number of the support, `support_type` is an integer between 1 and 8 identifying one of the eight supports (S1 to S8) shown in Fig. 8.2.5, and parameter a is an incline angle for supports S5 and S8, and zero for all other types of supports. Table 8.3.1 lists the entries of `Support_k` for these supports.

Note. The information of a frame set up by `setframe` can be accessed at any time by typing

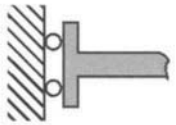
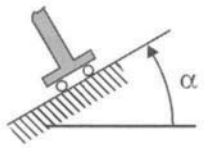
```
>> systinfo
```

TABLE 8.3.1 Specification of Supports

Support		Support_k
(S1) Fixed end $U = 0, V = 0, \theta = 0$		[node_no 1 0]
(S2) Pin or hinge $U = 0, V = 0$		[node_no 2 0]
(S3) Roller on a horizontal surface $V = 0$		[node_no 3 0]
(S4) Roller on a vertical surface $U = 0$		[node_no 4 0]
(S5) Roller on an incline of angle α $V_n = 0$		[node_no 5 α] Angle α is given in degrees.
(S6) Slider on a horizontal surface $V = 0, \theta = 0$		[node_no 6 0]

(continued)

TABLE 8.3.1 Specification of Supports (*Continued*)

Support		Support_k
(S7) Slider on a vertical surface $U = 0, \theta = 0$		[node_no 7 0]
(S8) Slider on an incline of angle α $V_n = 0, \theta = 0$		[node_no 8 α] Angle α is given in degrees.

EXAMPLE 3.1

Consider the frame in Fig. 8.3.1, which has a fixed end at node 1, a roller at node 2, and a hinged end at node 4. Let $a = 2$ m. Assume that every member has the same stiffness parameters $EA = 750$ kN, $EI = 1{,}000$ kN-m^2. The input arguments for `setframe` are

$$\texttt{Member\_Spec} = \begin{bmatrix} EA & EI & 1 & 3 \\ EA & EI & 2 & 3 \\ EA & EI & 3 & 4 \end{bmatrix}, \quad \texttt{Node\_Spec} = \begin{bmatrix} 0 & 0 \\ 2 & 0 \\ 2 & 2 \\ 4 & 2 \end{bmatrix},$$

$$\texttt{Support\_Spec} = \begin{bmatrix} 1 & 1 & 0 \\ 2 & 3 & 0 \\ 4 & 2 & 0 \end{bmatrix}.$$

The frame is set up by the following commands:

```
>> EA = 750; EI = 1000;
>> Member_Spec = [EA EI 1 3; EA EI 2 3; EA EI 3 4];
>> Node_Spec = [0 0; 2 0; 2 2; 4 2];
>> Support_Spec = [1 1 0; 2 3 0; 4 2 0];
>> setframe(Member_Spec, Node_Spec, Support_Spec);
```

which yield the undeformed configuration of the frame (not shown here), and the following information:

```
STATIC ANALYSIS OF A PLANE FRAME
================================

Description of the Frame
--------------------------------

Members (total number = 3)
 Member 1: EA = 750: EI = 1000, L = 2.8284, left node= 1, right node = 3
```

```
 Member 2: EA = 750: EI = 1000, L = 2, left node = 2,
           right node = 3
 Member 3: EA = 750: EI = 1000, L = 2, left node = 3,
           right node = 4

Here EA = longitudinal rigidity, EI = bending stiffness, L = length
 left node = node number of left end of the member
 right node = node number of right end of the member

Nodes (total number = 4)
 Node 1: X = 0, Y = 0
 Node 2: X = 2, Y = 0
 Node 3: X = 2, Y = 2
 Node 4: X = 4, Y = 2

Supports:
 at node 1, fixed end with U = 0, V = 0, theta = 0
 at node 2, roller placed on a horizontal surface, V = 0
 at node 4, hinge with U = 0, V = 0
```

8.3.2 Response to External Forces

External forces are divided into two categories: (i) distributed and concentrated loads that are applied at the interior points of frame members; and (ii) concentrated loads that are applied to the nodes of a frame. These loads are treated as follows:

- Concentrated loads applied at a node are described in the global coordinates (X, Y); see Fig. 8.3.3, where positive forces are in the directions of the X- and Y-axes, and positive torques are in the counterclockwise direction.
- External forces applied at the interior points of a member are described in the local coordinates (x, y) of the member; see Fig. 8.3.4, where positive forces are in the directions of the x- and y-axes.

The Toolbox of this chapter has a function `frameresp` for computing the static response of a frame under external forces; see Window 3.2.

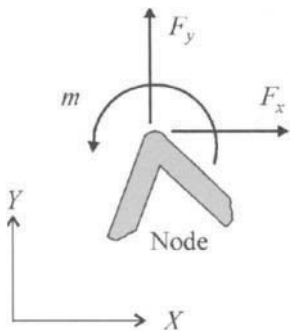


FIGURE 8.3.3 Concentrated loads at a node.

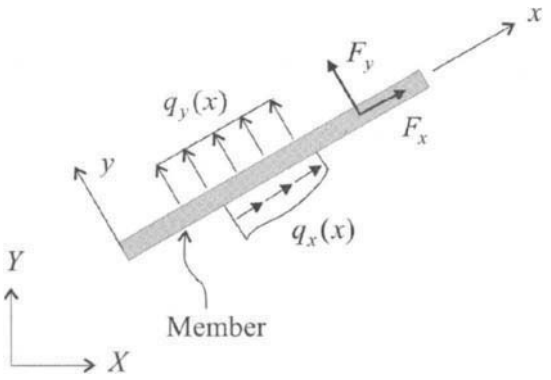


FIGURE 8.3.4 External loads applied at the interior of a member.

Window 3.2. Function frameresp

MATLAB Function: **frameresp**

Purpose:
To compute and display the static response of a frame subject to external loads.

Synopsis:

```
frameresp(Load_Int, Load_Ndl)
U_nodal = frameresp(Load_Int, Load_Ndl)
```

Description:
`frameresp(Load_Int, Load_Ndl)` determines the response of a frame subject to external loads, and plots the deformed and undeformed configurations of the frame. Here, `Load_Int` is a matrix describing external forces applied at the interior points of the frame members by Table 8.3.2, and `Load_Ndl` is a matrix describing concentrated loads applied at the nodes of the frame. These matrices can be replaced by `0` or `[ ]` if the corresponding type of loads does not exist. In other words, `frameresp(Load_Int)` determines the response to the forces applied to the interior of frame members only, while `frameresp(0, Load_Ndl)` obtains the response due to concentrated loads applied to nodes only.

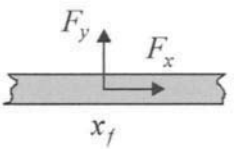
`U_nodal = frameresp(Load_Int, Load_Ndl)` returns the computed nodal displacements in a three-column matrix

$$\texttt{U\_nodal} = \begin{bmatrix} U_1 & V_1 & \theta_1 \\ U_2 & V_2 & \theta_2 \\ \vdots & \vdots & \vdots \\ U_N & V_N & \theta_N \end{bmatrix}$$

where U_1 and V_1 are the horizontal and vertical displacements of the ith node in the global coordinates X, Y, and θ_i is the rotation angle at the node that is measured in degrees and in the counterclockwise direction.

TABLE 8.3.2 External Loads Applied at the Interior of Frame Members

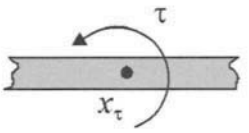
(L1) Pointwise force



$f_x(x) = F_x\delta(x - x_f), \quad f_y(x) = F_y\delta(x - x_f)$

`LD_int_k = [mem_no 0` x_q F_x F_y `0]`

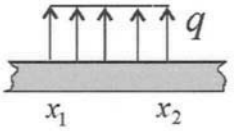
(L2) Torque



$f_y(x) = -\tau\dfrac{d}{dx}\delta(x - x_\tau)$

`LD_int_k = [mem_no -1` x_τ τ `0 0]`

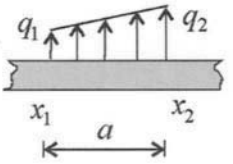
(D1) Uniformly distributed transverse force



$f_y(x) = q \quad \text{for } x_1 \le x \le x_2$

`LD_int_k = [mem_no 1` x_1 x_2 q `0]`

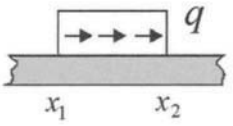
(D2) Linearly distributed transverse force



$f_y(x) = q_1 + \dfrac{q_2 - q_1}{a}(x - x_1) \quad \text{for } x_1 \le x \le x_2$

`LD_int_k = [mem_no 2` x_1 x_2 q_1 q_2`]`

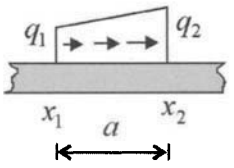
(D3) Uniformly distributed axial force



$f_x(x) = q \quad \text{for } x_1 \le x \le x_2$

`LD_int_k = [mem_no 3` x_1 x_2 q `0]`

(D4) Linearly distributed axial force



$f_x(x) = q_1 + \dfrac{q_2 - q_1}{a}(x - x_1) \quad \text{for } x_1 \le x \le x_2$

`LD_int_k = [mem_no 4` x_1 x_2 q_1 q_2`]`

The load specification matrices used by function `frameresp` are described as follows:

(i) External loads applied to the interior of frame members are specified by a matrix

$$\texttt{Load\_Int} = \begin{bmatrix} \texttt{LD\_Int\_1} \\ \texttt{LD\_Int\_2} \\ \vdots \\ \texttt{LD\_Int\_Nf1} \end{bmatrix}$$

A typical row of the matrix is

```
LD_int_k = [mem_no load_type p1 p2 p3 p4]
```

where `mem_no` is a member identification number, `load_type` is an integer between -1 and 4 identifying one of the six external loads considered in the Toolbox, and

p1 to p4 are parameters describing the location and form of the load. See Table 8.3.2 for the entries of LD_int_k.

(ii) External loads applied to the nodes of the frame are specified by a matrix

$$\text{Load\_Ndl} = \begin{bmatrix} \text{LD\_Ndl\_1} \\ \text{LD\_Ndl\_2} \\ \vdots \\ \text{LD\_\_Ndl\_Nf2} \end{bmatrix}$$

A typical row is of the form

```
LD_Ndl_k = [node_no Fx Fy m]
```

where node_no is a node number, Fx and Fy are the horizontal and vertical forces, and m is an external torque (see Fig. 8.3.3). A positive torque is in counterclockwise direction.

EXAMPLE 3.2

Consider the frame in Fig. 8.3.1. Let the parameters EA, EI, and a be the same as given in Example 3.1. Assume that the frame has been set up by function setframe. Let the external loads be $F = 80$ kN, which is applied to node 3, and $q = 50$ kN/m, which is applied to member 3. The corresponding force specification matrices are

```
Load_Int = [3 1 0 2 -50 0]
Load_Ndl = [3 -35 0 0]
```

The response of the frame under the external loads is computed by the commands

```
>> Load_Int = [3 1 0 2 -50 0];
>> Load_Ndl = [3 -80 0 0];
>> frameresp(Load_Int, Load_Ndl)
```

which yield the frame configurations in Fig. 8.3.5 and the following results:

```
Static Response of the Frame
----------------------------

INPUT DATA:

External forces applied at interior points of members
on member 3: Load D1 - uniform in transverse direction,
             range = [0 2], q0 = -50

External forces applied at nodes
  at node 3: Fx = -80, Fy = 0, Tau = 0

COMPUTED RESULTS:

Nodal displacements U, V and theta
  at node 1:  U = 0, V = 0, theta = 0 deg
  at node 2:  U = -0.088971, V = 0, theta = 1.1846 deg
```

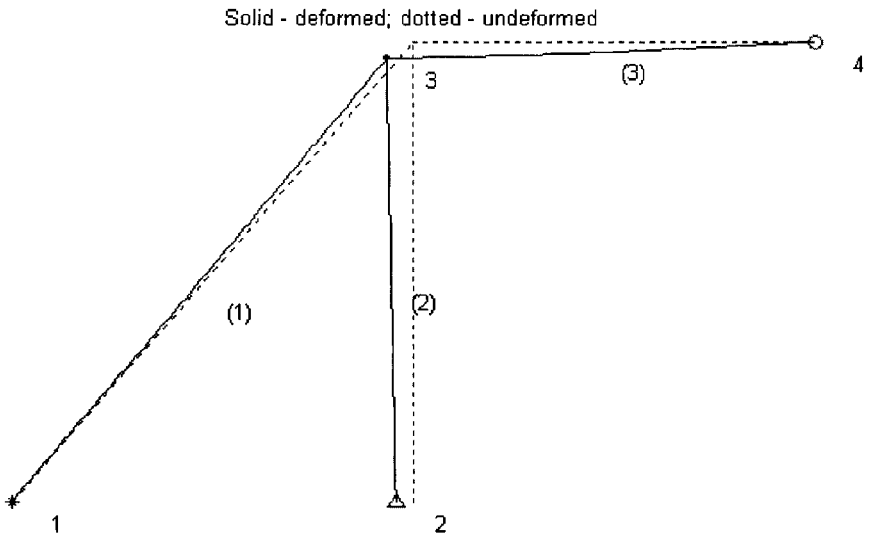


FIGURE 8.3.5

```
  at node 3:  U = -0.13032, V = -0.073465, theta = 1.1846 deg
  at node 4:  U = 0, V = 0, theta = 3.0421 deg

Reactions
  at node 1:  Rx = 31.1299
  at node 1:  Ry = 22.907
  at node 1:  Mc = -15.5325
  at node 2:  Ry = 27.5496
  at node 4:  Rx = 48.8701
  at node 4:  Ry = 49.5434

Maximum response of the frame in absolute value
  axial displacement    u_max = -0.1441,  on member 1, at x = 2.8284
  axial force        S_max = 48.8701,  on member 3, at x = 0
  transverse displacement w_max = 0.13032,  on member 2, at x = 2
  rotation         (dw/dx)_max = 0.053095,  on member 3, at x = 2
  bending moment      M_max = 24.5434,  on member 3, at x = 1
  shear force        Q_max = 50.4566,  on member 3, at x = 0
```

8.3.3 Response to Settlement of Supports

The supports of a frame may undergo deflections or displacements, which is referred to as *settlement of supports*. The support of a bridge undergoing a downward deflection is one example. Under support settlement, the configuration of a frame will change. For a statically indeterminate frame, this will cause elastic deformation and internal forces in the frame members. The function `frameresp` from the Toolbox can be used to compute the static response of a frame under support settlement; see Window 3.3.

Window 3.3. Function frameresp for Response Due to Support Settlement

MATLAB Function: **frameresp**

Purpose:
To compute and display the response of a frame due to settlement of supports.

Synopsis:

```
frameresp(0, 0, SD_Spec)
U_nodal_XY = frameresp(0, 0, SD_Spec)
```

Description:
`frameresp(0, 0, SD_Spec)` determines the response of a frame under settlement of supports, which is described by a matrix

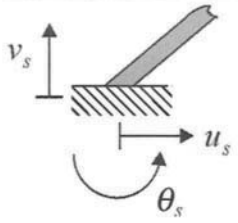
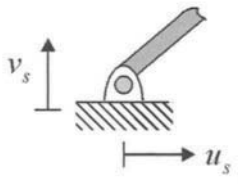
$$\texttt{SD\_Spec} = \begin{bmatrix} \texttt{SD\_1} \\ \texttt{SD\_2} \\ \vdots \\ \texttt{SD\_Ns} \end{bmatrix}$$

where each row of the matrix specifies a support settlement. A typical row of `SD_Spec` has the form

```
SD_k = [node_no support_type u1 u2 u3]
```

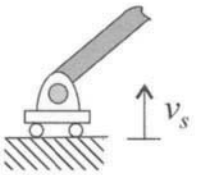
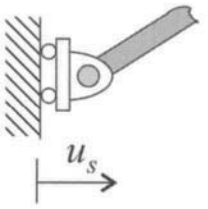
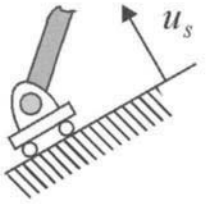
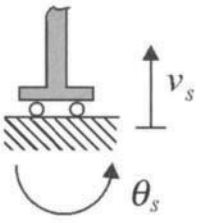
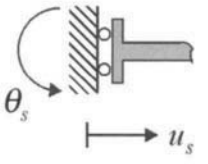
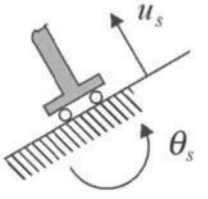
where `node_no` is a node identification number, `support_type` is an integer between 1 and 8 specifying one of the eight types of supports (S1 to S8), and parameters `u1` to `u3` describe appropriate support displacements. See Table 8.3.3 for the entries of `SD_k`.
`U_nodal_XY = frameresp(0, 0, SD_Spec)` returns the computed nodal displacements in a matrix `U_nodal`, as explained in Window 3.2.

TABLE 8.3.3 Specification of Support Settlement

Settlement of Support		SD_k
(S1) Fixed end $U = u_S, V = v_S$ $\theta = \theta_S$		[node_no 1 u_S v_S θ_S]
(S2) Pin or hinge $U = u_S, V = v_S$		[node_no 2 u_S v_S 0]

(continued)

TABLE 8.3.3 Specification of Support Settlement (*Continued*)

Settlement of Support		SD_k
(S3) Roller on a horizontal surface $V = v_s$		[node_no 3 0 v_s 0]
(S4) Roller on a vertical surface $U = u_s$		[node_no 4 u_s 0 0]
(S5) Roller on an incline of angle α $V_n = v_s$		[node_no 5 0 v_s 0]
(S6) Slider on a horizontal surface $V = v_s, \theta = \theta_s$		[node_no 6 0 v_s θ_s]
(S7) Slider on a vertical surface $U = u_s, \theta = \theta_s$		[node_no 7 u_s 0 θ_s]
(S8) Slider on an incline of angle α $V_n = v_s, \theta = \theta_s$		[node_no 8 0 v_s θ_s]

EXAMPLE 3.3

For the same frame in Example 3.1, assume that node 1 experiences a vertical displacement $v_s = -0.12$ m; see Fig. 8.3.6. The specification vector for the support settlement is

```
SD_Spec = [1 1 0 -0.12 0].
```

Assume that the structure has been set up following Example 3.1.

```
>> frameresp(0, 0, [1 1 0 -0.12 0])
```

plots the deformed frame in Fig. 8.3.7, and yields the following results:

```
COMPUTED RESULTS:

Nodal displacements U, V and theta
 at node 1:  U = 0, V = -0.12, theta = 0 deg
 at node 2:  U = 0.047201, V = 0, theta = 1.5146 deg
 at node 3:  U = -0.0056684, V = -0.0473, theta = 1.5146 deg
 at node 4:  U = 0, V = 0, theta = 1.2753 deg
```

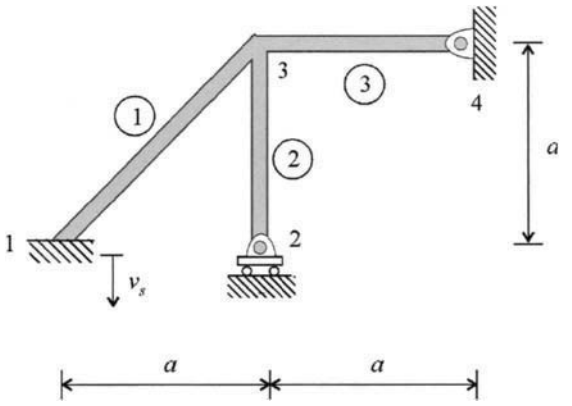


FIGURE 8.3.6

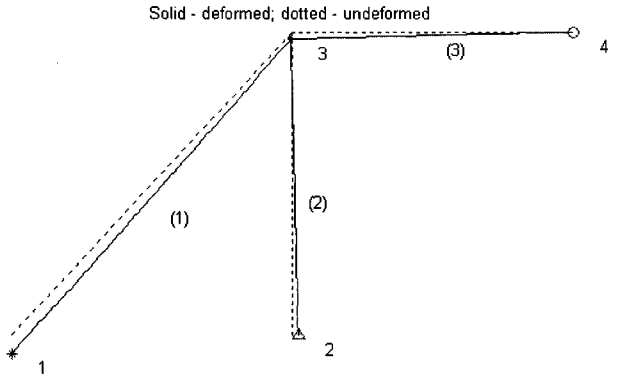


FIGURE 8.3.7

```
Reactions
 at node 1:  Rx = -2.1257
 at node 1:  Ry = -15.6488
 at node 1:  Mc = -22.8692
 at node 2:  Ry = 17.7374
 at node 4:  Rx = 2.1257
 at node 4:  Ry = -2.0886

Maximum response of the frame in absolute value
 axial displacement    u_max = -0.084853,  on member 1, at x = 0
 axial force       S_max = -17.7374,  on member 2, at x = 0
 transverse displacement w_max = -0.084853,  on member 1, at x = 0
 rotation         (dw/dx)_max = 0.027346, on member 1, at x = 2.4042
 bending moment      M_max = 22.8692,  on member 1, at x = 0
 shear force       Q_max = -9.5623,  on member 1, at x = 0
```

8.3.4 Total Response

When a frame is subject to both external loads and support settlement, the total response of the structure can be obtained by the principle of superposition. Function `frameresp` is capable of computing such total response; see Fig. 8.3.4.

Window 3.4. Function frameresp for Computing Total Response

MATLAB Function: **frameresp**

Purpose:
To compute the total response of a frame subject to external loads and support settlement.

Synopsis:

```
frameresp(Load_Int, Load_Ndl, SD_Spec)
U_nodal = frameresp(Load_Int, Load_Ndl, SD_Spec)
```

Description:
`frameresp(Load_Int, Load_Ndl, SD_Spec)` determines the total response of a frame subject to external loads described by matrices `Load_Int` and `Load_Ndl`, and support settlement described by a matrix `SD_Spec`. The specification of `Load_Int`, `Load_Ndl`, and `SD_Spec` has been given in Windows 3.2 and 3.3. Any one of these matrices can be replaced by 0 or [ ] as placeholder if the corresponding disturbance does not exist. For instance, for a frame subject to external loads applied to the interior points of frame members and settlement of supports, its response can be obtained by

```
frameresp(Load_Int, 0, SD_Spec)
```

`U_nodal = frameresp(Load_Int, Load_Ndl, SD_Spec)` returns the computed nodal displacements in a matrix `U_nodal`, as explained in Window 3.2.

EXAMPLE 3.4

In Fig. 8.3.8, a four-member frame is subject to a pointwise force ($F = 50$ kN) at node 2, a torque ($\tau = 80$ kN-m) on member 2, and a distributed load ($q = 5$ kN/m) on member 4. The frame is supported at nodes 1 and 4 as fixed ends, and at node 5 by a roller on an incline of angle $\alpha = 20°$. The support at node 4 has a vertical settlement $v_s = -0.08$ m. Let all the members have the same stiffness parameters: EA = 1,750 kN, EI = 2,000 kN-m^2. The total response of the frame is obtained by the following commands:

```
>> EA =1750; EI = 2000;
>> members = [EA EI 1 2; EA EI 2 3; EA EI 3 4; EA EI 3 5];
>> nodes = [0 0; 0 8; 10 8; 10 0; 18 0];
>> supports = [1 1 0; 4 1 0; 5 5 20];
>> setframe(members, nodes, supports);
>> qx = -5*cos(pi/4); qy = -5*sin(pi/4);
>> Lb = 8/sin(pi/4);          % length of member 4
>> ld_interior = [2 -1 5 80 0 0; 4 1 0 Lb qy 0; 4 3 0 Lb qx 0];
>> ld_nodal = [2 0 -50 0];
>> support_disp = [4 1 0 -0.08 0];
>> frameresp(ld_interior, ld_nodal, support_disp);
```

which yield the following results and the frame configurations in Fig. 8.3.9.

```
Nodal displacements U, V and theta
 at node 1: U = 0, V = 0, theta = 0 deg
 at node 2: U = -0.12093, V = -0.26741, theta = 0.79802 deg
 at node 3: U = -0.1384, V = -0.13885, theta = -0.99841 deg
 at node 4: U = 0, V = -0.08, theta = 0 deg
 at node 5 (a roller on an incline of angle = 20 deg):
           Ut (tangential) = -0.21262, Vn (normal) = 0,
           theta = 3.5804 deg

Reactions
 at node 1: Rx = 3.057
 at node 1: Ry = 58.495
```

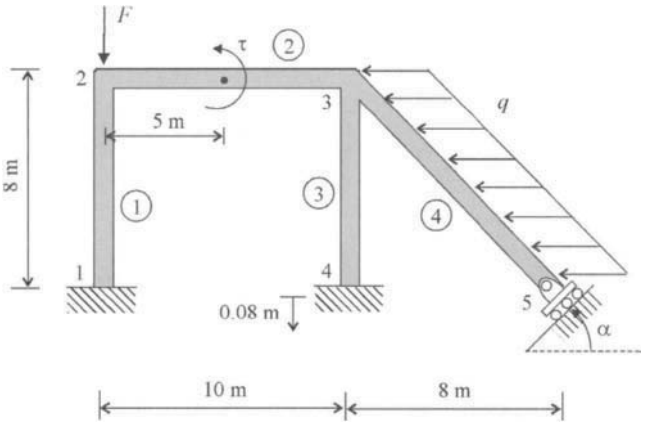


FIGURE 8.3.8

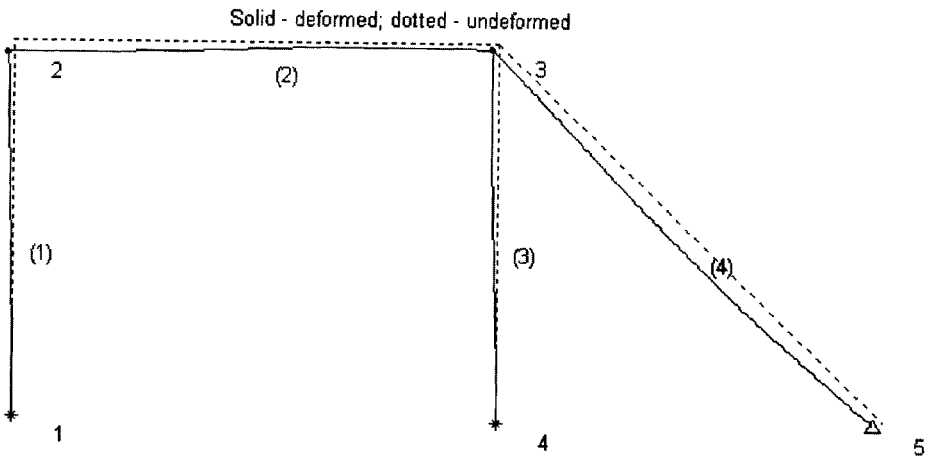


FIGURE 8.3.9

```
at node 1: Mc = -15.7099
at node 4: Rx = 9.7546
at node 4: Ry = 12.874
at node 4: Mc = -34.6621
at node 5 (a roller on an incline of angle = 20 deg):
   Rn (normal to surface) = 37.4586

Maximum response of the frame in absolute value
 axial displacement    u_max = -0.26741,  on member 1, at x = 8
 axial force         S_max = -58.495,  on member 1, at x = 0
 transverse displacement w_max = -0.38627, on member 4, at x = 6.3357
 rotation          (dw/dx)_max = 0.06249, on member 4, at x = 11.3137
 bending moment      M_max = -47.1705,  on member 4, at x = 0
 shear force         Q_max = 24.1693,  on member 4, at x = 0
```

After `frameresp` is executed, three functions from the Toolbox can be called to display the computed results; see Windows 3.5 to 3.7.

Window 3.5. Function `plotframe`

MATLAB Function: **`plotframe`**

Purpose:
To plot undeformed and deformed configurations of a frame.

Synopsis:

```
plotframe
plotframe (opt, display_ID)
```

Window 3.5. Function `plotframe` (*Continued*)

Description:
`plotframe` plots both deformed and undeformed configurations of a frame, in which a fixed end is expressed by a star (*), a hinge by a circle (o), and a roller or slider by a triangle (Δ).

`plotframe (opt, display_ID)` plots frame configurations with the following options:

`opt = 0`	plot undeformed configuration only
`opt = 1`	plot deformed configuration only
`opt = 2`	plot both deformed and undeformed configurations
`display_ID = 0`	hide member and node identification numbers
`display_ID = 1`	show member and node identification numbers

Member identification numbers are given in parentheses. By default, `display_ID = 0`.

Window 3.6. Function `maxresp`

MATLAB Function: **`maxresp`**

Purpose:
To display the maximum response of a frame in absolute value and locations.

Synopsis:

```
maxresp
```

Window 3.7. Function `reactf`

MATLAB Function: **`reactf`**

Purpose:
To obtain reactions at the supports of a frame.

Synopsis:

```
reactf
[Rc, D] = reactf
```

Description:
`reactf` displays the information of the reactions at the supports of a frame.

`[Rc, D] = reactf` returns the reactions in a column vector `Rc`, and the location and orientation of the reactions in a two-column matrix `D`. Here, the element `Rc(j)` is the

Window 3.7. Function `reactf` (*Continued*)

value of the jth reaction, the element `D(j,1)` is the identification number of the node at which the reaction `Rc(j)` acts, and the element `D(j,2)` tells the orientation of the reaction by the following options:

```
D(j,2) = 1   Rc(j) is a horizontal reaction force
D(j,2) = 2   Rc(j) is a vertical reaction force
D(j,2) = 3   Rc(j) is a reaction torque
```

For instance, `Rc(j) = 700`, `D(j,1) = 3`, and `D(j,2) = 2` indicate that a vertical reaction force of 700 force units is applied at node 3.

8.3.5 Response of Frame Members

Once the nodal displacements of a frame are obtained by function `frameresp`, the spatial distribution of internal forces and displacements of each member can be obtained in the local coordinates (x, y) of the member shown in Fig. 8.3.10. The displacements $u(x)$, $w(x)$, and internal forces $S(x)$, $M(x)$, and $Q(x)$ have been defined in Section 8.2.

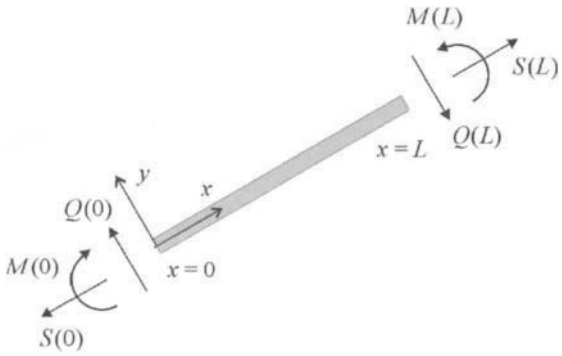


FIGURE 8.3.10 A frame member with internal forces at its ends.

The Toolbox of this chapter provides functions `membresp` and `plotmember` for evaluating and displaying the response of frame members; see Windows 3.8 and 3.9.

Window 3.8. Function `membresp`

MATLAB Function: **`membresp`**

Purpose:
To determine and display the displacements and internal forces of frame members in the local coordinates x, y.

Synopsis:

```
membresp
[y, x] = membresp
```

Window 3.8. Function membresp (*Continued*)

Description:
membresp displays the internal forces (S, M, Q) of all frame members at their ends.

[y, x] = membresp returns the computed response of frame members in a three-dimensional matrix y and the spatial points along the member axes that are used to computed the member response in a matrix x. For the kth member, its response can be retrieved from the matrix y as follows:

y(1, :, k) - axial displacement $u(x)$;
y(2, :, k) - axial force $S(x)$;
y(3, :, k) - transverse displacement w(x);
y(4, :, k) - rotation $\frac{d}{dx}$w(x) in radians;
y(5, :, k) - bending moment $M(x)$;
y(6, :, k) - shear force $Q(x)$.

The corresponding spatial points of simulation along the member are stored in x(k, :). For instance, plot(x(k, :), y(6, :, k)) plots the shear force distribution of the kth member.

Window 3.9. Function plotmember

MATLAB Function: **plotmember**

Purpose:
To plot the spatial distribution of the response of a selected frame member.

Synopsis:

```
plotmember(k)
plotmember(k, opt)
```

Description:
plotmember(k) plots the spatial distribution of the internal forces (S, M, Q) of the kth member and displays the response of the member at its two ends.

plotmember(k, opt) plots the response of the kth member with the following options:

opt = 1	plot internal forces S(x), M(x), Q(x)
opt = 2	plot displacements u(x), w(x)
opt = 3	plot axial response u(x), S(x)
opt = 4	plot transverse response w(x), dw(x)/dx, M(x), Q(x)
opt = 11	plot axial displacement u(x) only
opt = 12	plot axial force S(x) only
opt = 13	plot transverse displacement w(x) only
opt = 14	plot rotation dw(x)/dx only
opt = 15	plot bending moment M(x) only
opt = 16	plot shear force Q(x) only

EXAMPLE 3.5

Consider the frame in Example 3.4 for which the nodal displacements have been obtained by function `frameresp`.

```
>> membresp
```

gives the following information about the internal forces at the ends of the members:

Member		*S*	*M*	*Q*
Member 1	x = 0	−58.495	15.7099	−3.057
	x = *L*	−58.495	−8.7459	−3.057
Member 2	x = 0	−3.057	−8.7459	8.495
	x = *L*	−3.057	−3.7956	8.495
Member 3	x = 0	−12.874	43.3749	−9.7546
	x = *L*	−12.874	−34.6621	−9.7546
Member 4	x = 0	6.051	−47.1705	24.1693
	x = *L*	−33.949	0	−15.8307

where S is the axial force, M the bending moment, and Q the shear force. To plot the spatial distribution of the transverse response of member 4, type

```
>> plotmember(4,4)
```

which yields Fig. 8.3.11.

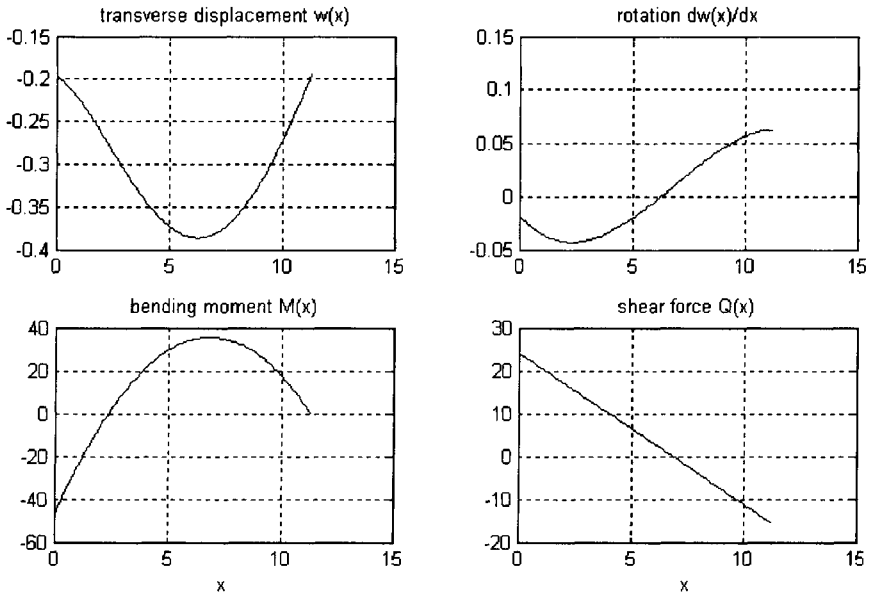


FIGURE 8.3.11 Transverse response of member 4 of the frame in Fig. 8.3.8.

8.4 The Distributed Transfer Function Method

This section presents the basic concepts about the Distributed Transfer Function Method (DTFM), which is used to develop the Toolbox for frame analysis. The user who is only interested in using the Toolbox for computation can skip this section.

The DTFM was originally developed for vibration analysis and control of distributed dynamic systems. It turns out that the method is also efficient in static analysis of structures, including plane frames. For this reason, we keep the words transfer function in the name, although the method actually is a Green's function method for static problems. For more information on the DTFM, refer to Appendix C and Reference 5 in Section 8.6.

State Formulation for Frame Members

In a DTFM-based analysis, Eqs. (8.1) and (8.3) are cast in spatial state form

$$\frac{d}{dx}\{\eta_1(x)\} = [F_1]\{\eta_1(x)\} - \frac{1}{EA} f_x(x)\{e_2\} \tag{8.9a}$$

$$\frac{d}{dx}\{\eta_2(x)\} = [F_2]\{\eta_2(x)\} + \frac{1}{EI} f_y(x)\{e_4\} \tag{8.9b}$$

where the state vectors given by

$$\begin{aligned} \{\eta_1(x)\} &= \left(u(x) \quad \frac{d}{dx}u(x)\right)^T \\ \{\eta_2(x)\} &= \left(w(x) \quad \frac{d}{dx}w(x) \quad \frac{d^2}{dx^2}w(x) \quad \frac{d^3}{dx^3}w(x)\right)^T \end{aligned} \tag{8.10a}$$

the state matrices are given by

$$[F_1] = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad [F_2] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{8.10b}$$

and $\{e_2\} = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$, $\{e_4\} = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T$. The boundary conditions for a member connected to nodes i and j are

$$\text{for Eq. (8.9a):} \qquad [M_1]\{\eta_1(0)\} + [N_1]\{\eta_1(L)\} = \{\gamma_1\} \tag{8.11a}$$

$$\text{for Eq. (8.9b):} \qquad [M_2]\{\eta_2(0)\} + [N_2]\{\eta_2(L)\} = \{\gamma_2\} \tag{8.11b}$$

with

$$[M_1] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad [N_1] = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \{\gamma_1\} = \begin{pmatrix} u_i \\ u_j \end{pmatrix} \tag{8.12a}$$

$$[M_2] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad [M_2] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \{\gamma_2\} = \begin{pmatrix} w_i \\ \theta_i \\ w_j \\ \theta_j \end{pmatrix} \tag{8.12b}$$

where u_i, w_i, θ_i, u_j, w_j, and θ_j are the nodal displacements defined in Fig. 8.2.4(a).

Member Stiffness Matrix and Transmitted Forces

By the DTFM, the solutions of Eqs. (8.9) with the boundary conditions (8.11) are

$$\{\eta_1(x)\} = -\frac{1}{EA}\int_0^L G_1(x,\xi) f_x(\xi) d\xi \, \{e_2\} + H_1(x)\{\gamma_1\} \tag{8.13a}$$

$$\{\eta_2(x)\} = \frac{1}{EI}\int_0^L G_2(x,\xi) f_y(\xi) d\xi \, \{e_4\} + H_2(x)\{\gamma_2\} \tag{8.13b}$$

where $G_\alpha(x,\xi)$ and $H_\alpha(x)$, $\alpha = 1, 2$, are the Green's functions and boundary influence functions of the member, respectively. These functions are of the form

$$G_\alpha(x,\xi) = \begin{cases} H_\alpha(x)[M_\alpha] e^{-[F_\alpha]\xi} & \text{for} \quad x \geq \xi \\ -H_\alpha(x)[N_\alpha] e^{[F_\alpha](L-\xi)} & \text{for} \quad x \leq \xi \end{cases}$$
$$H_\alpha(x) = e^{[F_\alpha]x}\left([M_\alpha] + [N_\alpha] e^{[F_\alpha]L}\right)^{-1} \tag{8.14}$$

for $\alpha = 1, 2$, where the exponential matrices are

$$e^{[F_1]x} = \begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix}, \quad e^{[F_2]x} = \begin{bmatrix} 1 & x & x^2/2 & x^3/6 \\ 0 & 1 & x & x^2/2 \\ 0 & 0 & 1 & x \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.15}$$

Note that $G_\alpha(x,\xi)$ and $H_\alpha(x)$ are in exact and closed form.

With Eqs. (8.13), the internal forces of the member are expressed as

$$S(x) = \begin{bmatrix} 0 & EA \end{bmatrix}\{\eta_1(x)\}, \quad \begin{pmatrix} M(x) \\ Q(x) \end{pmatrix} = \begin{bmatrix} 0 & 0 & EI & 0 \\ 0 & 0 & 0 & EI \end{bmatrix}\{\eta_2(x)\} \tag{8.16}$$

Denote the elements of G_α and H_α by $g_{ij}^{(\alpha)}$ and $h_{ij}^{(\alpha)}$, respectively. Substitute Eq. (8.16) into Eq. (8.4) to obtain the forces of the member at nodes i and j

$$\{p_e\} = \left[\tilde{k}_e\right]\{u_e\} + \{\tilde{q}_e\} \tag{8.17}$$

where $\{p_e\}$ and $\{u_e\}$, the nodal force and displacement vectors, are given by

$$\{p_e\} = \begin{pmatrix} p_{xi} & p_{yi} & p_{mi} & p_{xj} & p_{yj} & p_{mj} \end{pmatrix}^T$$
$$\{u_e\} = \begin{pmatrix} u_i & w_i & \theta_i & u_j & w_j & \theta_j \end{pmatrix}^T \tag{8.18}$$

$\left[\tilde{k}_e\right]$ is a six-by-six stiffness matrix of the member of the form

$$\left[\tilde{k}_e\right] = \begin{bmatrix} -EAh_{21}^{(1)}(0) & 0 & 0 & -EAh_{22}^{(1)}(0) & 0 & 0 \\ 0 & EIh_{41}^{(2)}(0) & EIh_{42}^{(2)}(0) & 0 & EIh_{43}^{(2)}(0) & EIh_{44}^{(2)}(0) \\ 0 & -EIh_{31}^{(2)}(0) & -EIh_{32}^{(2)}(0) & 0 & -EIh_{33}^{(2)}(0) & -EIh_{34}^{(2)}(0) \\ EAh_{21}^{(1)}(L) & 0 & 0 & EAh_{22}^{(1)}(L) & 0 & 0 \\ 0 & -EIh_{41}^{(2)}(L) & -EIh_{42}^{(2)}(L) & 0 & -EIh_{43}^{(2)}(L) & -EIh_{44}^{(2)}(L) \\ 0 & EIh_{31}^{(2)}(L) & EIh_{32}^{(2)}(L) & 0 & EIh_{33}^{(2)}(L) & EIh_{34}^{(2)}(L) \end{bmatrix} \tag{8.19}$$

and $\{\tilde{q}_e\}$ is a vector of transmitted nodal forces due to the external forces $f_x(x)$ and $f_y(x)$ that are applied to the interior of the member. It can be shown that

$$\{\tilde{q}_e\} = \begin{pmatrix} \hat{q}_{xi} & \hat{q}_{yi} & \hat{q}_{mi} & \hat{q}_{xj} & \hat{q}_{yj} & \hat{q}_{mj} \end{pmatrix}^T \tag{8.20}$$

where

$$\hat{q}_{xi} = \int_0^L g_{22}^{(1)}(0,\xi) f_x(\xi)\, d\xi, \quad \hat{q}_{yi} = \int_0^L g_{44}^{(2)}(0,\xi) f_y(\xi)\, d\xi, \quad \hat{q}_{mi} = -\int_0^L g_{34}^{(2)}(0,\xi) f_y(\xi)\, d\xi$$

$$\hat{q}_{xj} = -\int_0^L g_{22}^{(1)}(L,\xi) f_x(\xi)\, d\xi, \quad \hat{q}_{yj} = -\int_0^L g_{44}^{(2)}(L,\xi) f_y(\xi)\, d\xi, \quad \hat{q}_{mj} = \int_0^L g_{34}^{(2)}(L,\xi) f_y(\xi)\, d\xi \tag{8.21}$$

The integrals in Eq. (8.21) can be obtained by exact quadrature for different forcing functions.

Coordinate Transformation

To analyze the balance of nodal forces at a node, a global coordinate system XY is established; see Fig. 8.4.1, where a member is inclined with an angle ϕ. The transformation of nodal displacements and forces from the local coordinates (x, y) to the global coordinates (X,Y) is shown in Fig. 8.4.2, where U_i and V_i are the global displacements at node i, and Q_{Xi} and

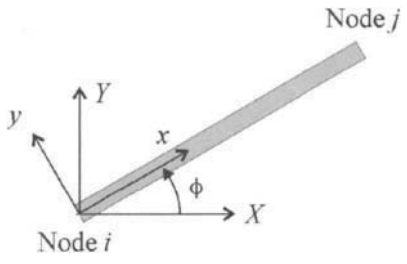


FIGURE 8.4.1 A member with local and global coordinates.

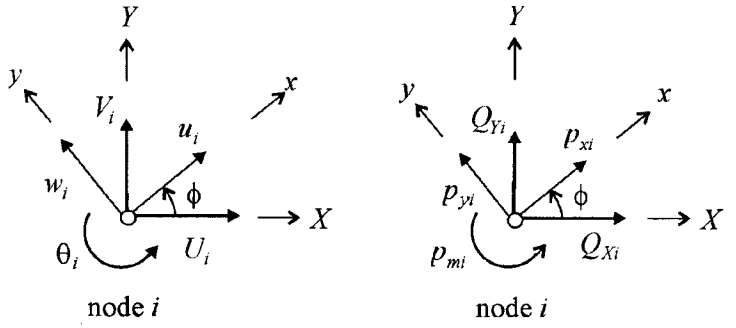


FIGURE 8.4.2 Global nodal displacements and forces.

Q_{Yi} are the global nodal forces. The rotation θ_i and nodal torque p_{mi} remain the same in both coordinate systems. It follows that

$$\begin{pmatrix} u_i \\ w_i \\ \theta_i \end{pmatrix} = [T_c] \begin{pmatrix} U_i \\ V_i \\ \theta_i \end{pmatrix}, \quad \begin{pmatrix} Q_{Xi} \\ Q_{Yi} \\ M_i \end{pmatrix} = [T_c]^T \begin{pmatrix} p_{xi} \\ p_{yi} \\ p_{mi} \end{pmatrix} \tag{8.22}$$

where the coordinate transformation matrix

$$[T_c] = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.23}$$

For node j, a similar coordinate transformation for nodal forces and displacements can be obtained. Apply Eq. (8.22) to Eq. (8.17) to obtain the relation between global nodal forces and global nodal displacements

$$\begin{pmatrix} Q_{Xi} \\ Q_{Yi} \\ M_i \\ Q_{Xj} \\ Q_{Yj} \\ M_j \end{pmatrix} = [k_e] \begin{pmatrix} U_i \\ V_i \\ \theta_i \\ U_j \\ V_j \\ \theta_j \end{pmatrix} + [T_e]^T \{\tilde{q}_e\} \tag{8.24}$$

where $[k_e]$ is the member stiffness matrix in global coordinates, and is given by

$$[k_e] = [T_e]^T \left[\tilde{k}_e\right] [T_e] \quad \text{with } [T_e] = \begin{bmatrix} [T_c] & [0] \\ [0] & [T_c] \end{bmatrix}. \tag{8.25}$$

Assembly of Frame

The frame is assembled from its members through force balance at all the nodes. A schematic of such member assembly is shown in Fig. 8.4.3, where members A, B, and C are rigidly connected at node i. The equilibrium equations at the node are

$$\begin{aligned} F_{Xi} - Q_{Xi}^{(A)} - Q_{Xi}^{(B)} - Q_{Xi}^{(C)} &= 0 \\ F_{Yi} - Q_{Yi}^{(A)} - Q_{Yi}^{(B)} - Q_{Yi}^{(C)} &= 0 \\ m_i - M_i^{(A)} - M_i^{(B)} - M_i^{(C)} &= 0 \end{aligned} \tag{8.26}$$

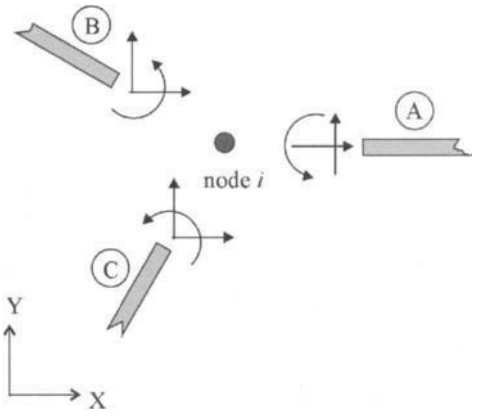


FIGURE 8.4.3 Force balance at a node.

where $Q_{Xi}^{(A)}$, $Q_{Yi}^{(A)}$, and $M_i^{(A)}$ are nodal forces of member A at node i in the global coordinates, as given in Eq. (8.22); F_{Xi}, F_{Yi}, and m_i are concentrated external loads applied at the node (not shown in Fig. 8.4.3). Suppose that the other ends of members A, B, and C are connected to nodes j, k, and l, respectively. According to Eq. (8.24), Eq. (8.26) is reduced to

$$\begin{aligned}&\left[k_A^{(i)}\right]\begin{pmatrix}U_i\\V_i\\\theta_i\end{pmatrix}+\left[k_B^{(i)}\right]\begin{pmatrix}U_i\\V_i\\\theta_i\end{pmatrix}+\left[k_C^{(i)}\right]\begin{pmatrix}U_i\\V_i\\\theta_i\end{pmatrix}+\left[k_A^{(j)}\right]\begin{pmatrix}U_j\\V_j\\\theta_j\end{pmatrix}\\&+\left[k_B^{(l)}\right]\begin{pmatrix}U_k\\V_k\\\theta_k\end{pmatrix}+\left[k_C^{(m)}\right]\begin{pmatrix}U_l\\V_l\\\theta_l\end{pmatrix}=\begin{pmatrix}F_{Xi}\\F_{Yi}\\m_j\end{pmatrix}-\begin{pmatrix}q_{Xi}\\q_{Yi}\\q_{mj}\end{pmatrix}\end{aligned}\tag{8.27}$$

where U_i, V_i, and θ_i are the nodal displacements at node i; q_{Xi}, q_{Yi}, and q_{mj} are the resultants of transmitted nodal forces due to external forces applied to the interior of the members, which are given in Eq. (8.21); and $\left[k_A^{(i)}\right],\ldots,\left[k_C^{(m)}\right]$ are submatrices obtained from partition of the member stiffness matrices given in Eq. (8.25).

Force balance at all the nodes of the frame leads to a global equilibrium equation

$$[K]\{U\}=\{F\}\tag{8.28}$$

where $\{U\}$ is a vector of all nodal displacements; $\{F\}$ is a vector consisting of external loads and transmitted forces; and $[K]$ is the global stiffness matrix of the frame that is assembled from those submatrices like $\left[k_A^{(i)}\right],\ldots,\left[k_C^{(m)}\right]$ in Eq. (8.27).

Determination of Frame Response

To solve Eq. (8.28), the conditions at the supports of the frame must be introduced to eliminate dependent nodal displacements. For instance, if node i is a hinge (S2), $U_i = 0$, $V_i = 0$; and if node j is a roller on a horizontal surface which has support settlement (S3 in Table 8.3.3), $V_j = v_s$ where v_s is a known support displacement. Elimination of all known displacements at the supports from $\{U\}$, the corresponding entries from $\{F\}$, and the corresponding rows and columns from $[K]$, yields a modified equilibrium equation

$$\left[\hat{K}\right]\left\{\hat{U}\right\}=\left\{\hat{F}\right\}\tag{8.29}$$

where $\{\hat{U}\}$ only contains independent nodal displacements. Thus,

$$\{\hat{U}\} = [\hat{K}]^{-1}\{\hat{F}\}. \tag{8.30}$$

Substitution of $\{\hat{U}\}$ into Eqs. (8.13), (8.22), and (8.28) eventually leads to the static response of every member and the reactions of the frame.

The above-mentioned DTFM analysis is convenient in computer coding, and gives exact solutions for general external loads.

8.5 Quick Solution Guide

System Description

A frame structure is an assembly of a number of rigidly connected prismatic members; see Fig. 8.5.1. These members can undergo both longitudinal and transverse deformations. External loads can be applied at nodes and the interior of a member.

The longitudinal displacement $u(x)$ and transverse displacement $w(x)$ of a member, as shown in Fig. 8.5.2, are governed by the differential equations

$$-EA\frac{d^2}{dx^2}u(x) = f_x(x), \quad 0 \le x \le L$$

$$EI\frac{d^4}{dx^4}w(x) = f_y(x), \quad 0 \le x \le L$$

where EA and EI are the longitudinal rigidity and bending stiffness of the member. The internal forces of the member are

axial force	$S(x) = EA\dfrac{d}{dx}u(x)$
bending moment	$M(x) = EI\dfrac{d^2}{dx^2}w(x)$
shear force	$Q(x) = EI\dfrac{d^3}{dx^3}w(x)$

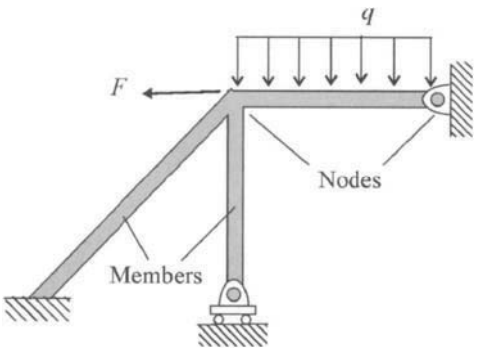


FIGURE 8.5.1 Schematic of a frame structure.

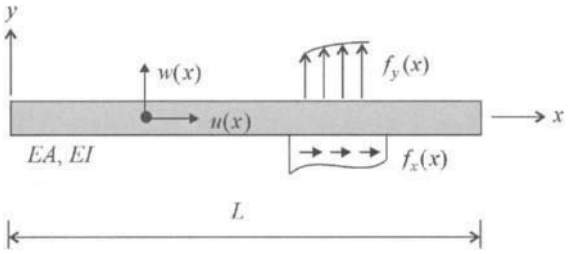


FIGURE 8.5.2 The displacements of a frame member.

Problems to Solve

This Toolbox has functions for determining the solutions of the following problems of frames:

(a) Static response to external loads; and
(b) Static response to support settlement.

MATLAB Functions

This chapter contains a toolbox of MATLAB functions for static analysis of plane frames; see the table below. Refer to the corresponding window or section for the utility of each function. The application of some major functions is summarized in the following pages.

Static Analysis of Frames		
`setframe`	Set up parameters and boundary conditions of a frame	Window 3.1
`frameresp`	Compute frame response subject to external loads	Window 3.2
	Compute frame response due to support settlement	Window 3.3
	Compute the total response of a frame	Window 3.4
`maxresp`	Obtain maximum response of a frame and locations	Window 3.5
`reactf`	Obtain reactions at supports	Window 3.6
`plotframe`	Plot undeformed and deformed configurations of a frame	Window 3.7
`membresp`	Compute response of frame members	Window 3.8
`plotmember`	Plot response along a frame member, and show response at the two ends of the member	Window 3.9
Utilities		
`systinfo`	Display frame parameters and boundary conditions	Window 3.1
`TBdemo`	Show how the Toolbox works and what it can do	Section 8.1
`Tbinfo`	Show the information of the Toolbox	Section 8.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 8.1

Solution Procedure

Computation and display of the static response of a frame takes the following three steps:

Step 1. Set up frame parameters and boundary conditions

```
>> setframe(Member_Spec, Node_Spec, Support_Spec)
```

Step 2. Compute static response

To compute response subject to external forces, type

```
>> frameresp(Load_Int, Load_Ndl)
```

To compute response due to settlement of supports, type

```
>> frameresp (0, 0, SD_Spec)
```

For total response, type

```
>> frameresp(Load_Int, Load_Ndl, SD_Spec)
```

Step 3. Plot response of frame members

```
>> plotmember(mem_no, opt)
```

where `mem_no` is member identification number, and

`opt` = 1 plot distributions of internal forces S(x), M(x), Q(x)
`opt` = 2 plot distribution of displacements w(x), u(x).

Note. Matrices `Member_Spec`, `Node_Spec`, and `Support_Spec` specify the parameters, nodes, and supports of the frame structure, respectively. Matrix `Load_Int` specifies external loads applied to the interior of members; `Load_Ndl` specifies concentrated external loads applied at nodes; and `SD_Spec` specifies settlement of supports. Specification of these matrices is given as follows.

Specification of Frame Members

$$\texttt{Member\_Spec} = \begin{bmatrix} EA_1 & EI_1 & n_{L1} & n_{R1} \\ EA_2 & EI_2 & n_{L2} & n_{L2} \\ \vdots & \vdots & \vdots & \vdots \\ EA_M & EI_M & n_{LM} & n_{LM} \end{bmatrix}$$

where n_{Li} and n_{Ri} are the node numbers of the left and right ends of the member, which also define the positive direction of the local coordinate x; see Fig. 8.5.3.

Specification of Nodes

$$\texttt{Node\_Spec} = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ \vdots & \vdots \\ X_N & Y_N \end{bmatrix}$$

where the ith row of the matrix contains the global coordinates of the ith node.

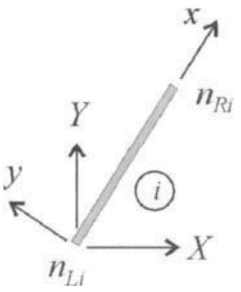
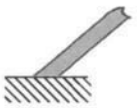


FIGURE 8.5.3 The node numbers and local coordinates of a member.

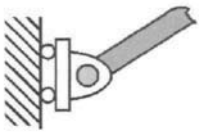
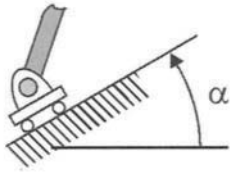
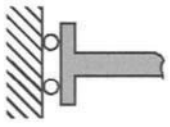
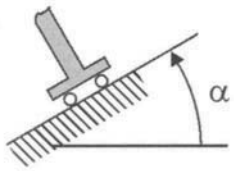
Specification of Supports

$$\texttt{Support\_Spec} = \begin{bmatrix} \texttt{Support\_1} \\ \texttt{Support\_2} \\ \vdots \\ \texttt{Support\_Ns} \end{bmatrix}$$

where each row specifies a support; see the following table, which is the same as Table 8.3.1. (The parameter `node_no` in the table is a node identification number.)

Support	Support_k
(S1) Fixed end $U = 0, V = 0, \theta = 0$ 	[node_no 1 0]
(S2) Pin or hinge $U = 0, V = 0$	[node_no 2 0]
(S3) Roller on a horizontal surface $V = 0$	[node_no 3 0]

(continued)

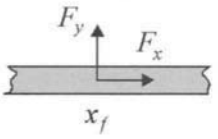
Support	Support_k
(S4) Roller on a vertical surface $U = 0$ 	[node_no 4 0]
(S5) Roller on an incline of angle α $V_n = 0$ 	[node_no 5 α] Angle α is given in degrees.
(S6) Slider on a horizontal surface $V = 0, \theta = 0$ 	[node_no 6 0]
(S7) Slider on a vertical surface $U = 0, \theta = 0$ 	[node_no 7 0]
(S8) Slider on an incline of angle α $V_n = 0, \theta = 0$ 	[node_no 8 α] Angle α is given in degrees.

Specification of External Loads Applied to the Interior of Frame Members

$$\texttt{Load\_Int} = \begin{bmatrix} \texttt{Load\_Int\_1} \\ \texttt{Load\_Int\_2} \\ \vdots \\ \texttt{Load\_Int\_Nf1} \end{bmatrix}$$

where each row specifies a load; see the following table, which is the same as Table 8.3.2.

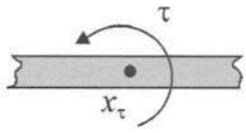
(L1) Pointwise force



$f_x(x) = F_x\delta(x - x_f), \quad f_y(x) = F_y\delta(x - x_f)$

`LD_int_k = [mem_no 0 x_q F_x F_y 0]`

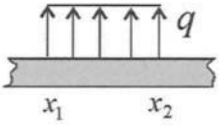
(L2) Torque



$f_y(x) = -\tau\dfrac{d}{dx}\delta(x - x_\tau)$

`LD_int_k = [mem_no -1 x_τ τ 0 0]`

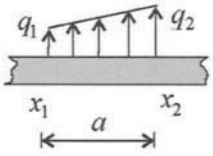
(D1) Uniformly distributed transverse force



$f_y(x) = q \quad \text{for } x_1 \le x \le x_2$

`LD_int_k = [mem_no 1 x_1 x_2 q 0]`

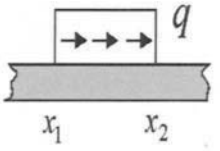
(D2) Linearly distributed transverse force



$f_y(x) = q_1 + \dfrac{q_2 - q_1}{a}(x - x_1) \quad \text{for } x_1 \le x \le x_2$

`LD_int_k = [mem_no 2 x_1 x_2 q_1 q_2]`

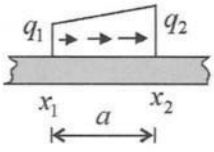
(D3) Uniformly distributed axial force



$f_x(x) = q \quad \text{for } x_1 \le x \le x_2$

`LD_int_k = [mem_no 3 x_1 x_2 q 0]`

(D4) Linearly distributed axial force



$f_x(x) = q_1 + \dfrac{q_2 - q_1}{a}(x - x_1) \quad \text{for } x_1 \le x \le x_2$

`LD_int_k = [mem_no 4 x_1 x_2 q_1 q_2]`

Specification of External Loads

$$\texttt{Load\_Ndl} = \begin{bmatrix} \texttt{Load\_Ndl\_1} \\ \texttt{Load\_Ndl\_2} \\ \vdots \\ \texttt{Load\_\_Ndl\_Nf2} \end{bmatrix}$$

The *k*th row is of the form

```
Load_k = [node_no Fx Fy m]
```

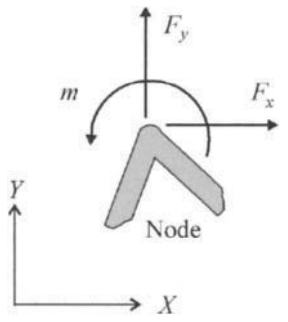


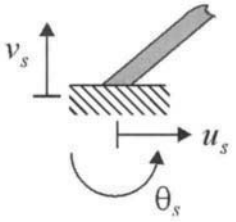
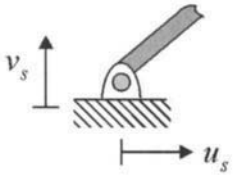
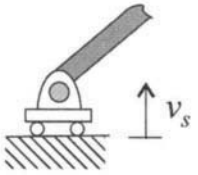
FIGURE 8.5.4 Concentrated loads at a node.

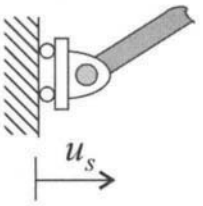
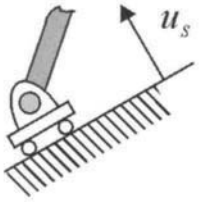
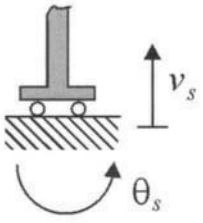
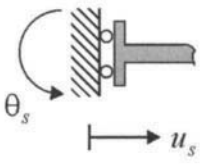
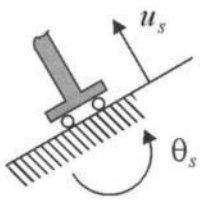
where `node_no` is a node identification number, `Fx` and `Fy` are the horizontal and vertical forces, and `m` is an external torque; see Fig. 8.5.4. The positive direction of `m` is counterclockwise.

Specification of Displacements at Supports (Settlement of Supports)

$$\texttt{SD\_Spec} = \begin{bmatrix} \texttt{SD\_1} \\ \texttt{SD\_2} \\ \vdots \\ \texttt{SD\_Ns} \end{bmatrix}$$

where each row of the matrix specifies a support displacement, according to the following table, which is the same as Table 8.3.3. (The parameter `node_no` in the table is a node identification number.)

Settlement of Support		`SD_k`
(S1) Fixed end $U = u_s,\ V = v_s$ $\theta = \theta_s$		`[node_no 1` u_s v_s θ_s`]`
(S2) Pin or hinge $U = u_s, V = v_s$		`[node_no 2` u_s v_s `0]`
(S3) Roller on a horizontal surface $V = v_s$		`[node_no 3 0` v_s `0]`

Settlement of Support	SD_k
(S4) Roller on a vertical surface $U = u_s$ 	[node_no 4 u_s 0 0]
(S5) Roller on an incline of angle α $V_n = v_s$ 	[node_no 5 0 v_s 0]
(S6) Slider on a horizontal surface $V = v_s, \theta = \theta_s$ 	[node_no 6 0 v_s θ_s]
(S7) Slider on a vertical surface $U = u_s, \theta = \theta_s$ 	[node_no 7 u_s 0 θ_s]
(S8) Slider on an incline of angle α $V_n = v_s, \theta = \theta_s$ 	[node_no 8 0 v_s θ_s]

8.6 References

1. Hibbeler R C 2001 *Structural Analysis*, 5th ed., Prentice-Hall: Upper Saddle River, New Jersey.
2. Kassimali A 1999 *Matrix Analysis of Structures*, Brooks Cole.
3. Przemienieck J S 1985 *Theory of Matrix Structural Analysis*, Dover Publications.
4. West H H, Geschwindner L F 2002 *Fundamentals of Structural Analysis*, 2nd ed., Wiley Text Books.
5. Yang B 1994 "Distributed Transfer Function Analysis of Complex Distributed Parameter Systems," *ASME Journal of Applied Mechanics*, Vol. 61. No. 1, pp. 84–92.

PART III
Dynamics and Vibrations

This Page Intentionally Left Blank

9 Dynamics of Particles and Rigid Bodies

Inside
- Getting Started
- Dynamics of Particles
- Dynamics of Rigid Bodies in Plane Motion
- Rigid Body Dynamics in Three Dimensions
- Quick Solution Guide
- References

9.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for dynamics of particles and rigid bodies.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

For a quick solution by the Toolbox, the reader may refer directly to the Quick Solution Guide (Section 9.5), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To better understand dynamics of particles and rigid bodies, the reader is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 9.6.

9.2 Dynamics of Particles

9.2.1 Preliminaries

Units Some commonly used quantities in engineering mechanics are listed in Table 9.2.1, where units in both the Standard International (SI) and U.S. Customary Systems are shown.

Newton's Laws Classical mechanics was established based upon the following three laws of motion that were first stated by Isaac Newton in 1687:

First Law A particle remains at rest or moves with a constant velocity along a straight line if the sum of the forces acting on it is zero.

Second Law The rate of change of the linear momentum of a particle is equal to the sum of the forces acting on it.

Third Law The forces exerted by two particles upon each other, which are called the action and reaction forces, are equal in magnitude, opposite in direction, and collinear.

Mathematically, Newton's second law is expressed by

$$\boldsymbol{F} = \frac{d}{dt}(m\boldsymbol{v}) \tag{9.1}$$

where m and $\boldsymbol{v}$ are the mass and velocity of the particle, respectively; $\boldsymbol{F}$ is the resultant or sum of all forces acting on the particle; and $m\boldsymbol{v}$ is the linear momentum of the particle. If the mass m is constant, Eq. (9.1) becomes

$$\boldsymbol{F} = m\boldsymbol{a} \tag{9.2}$$

TABLE 9.2.1 Units

Quantity	*SI Unit*	*U.S. Unit*
time	second (s)	second (sec)
length	meter (m)	foot (ft)
mass	kilogram (kg)	slug
force	Newton = kg·m/s^2 (N)	pound (lb)
moment of force	N·m	lb-ft
energy or work	N·m or joule (J)	ft-lb
impulse	N·s	lb-sec
momentum	kg·m/s	slug-ft/sec

Conversion:

1 ft = 0.3048 m; 1 lb = 4.4482 N; 1 slug = 14.594 kg

where $\boldsymbol{a} = \frac{d\boldsymbol{v}}{dt}$ is the acceleration of the particle. Equation (9.2) is a commonly used expression of Newton's second law, which indicates that the acceleration of a particle of constant mass is proportional to the resultant force.

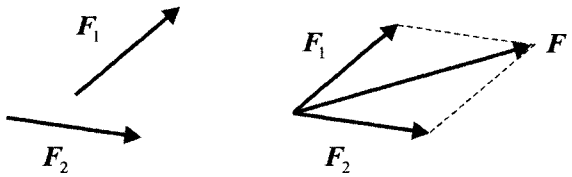
Besides Newton's laws of motion, Newton's law of gravitation is also important in dynamics, especially in orbital dynamics; see Sec. 9.2.6.

Force The motion of a particle or body is caused by forces. A force is an action of pull or push applied by one body to another. A force is characterized by its magnitude, direction of action, and point of application. Thus, forces can be expressed by vectors, and superposition of forces follows the parallelogram rule of vector summation. For instance, the sum of forces $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ in Fig. 9.2.1(a) is obtained by the diagonal of the parallelogram formed by the forces, as shown in Fig. 9.2.1(b).

There are various types of forces. Shown in Fig. 9.2.2 are three examples. In Fig. 9.2.2(a), the dry (Coulomb) friction F_r between a moving body and the supporting surface is proportional to the normal force N, where μ is the coefficient of kinetic friction. In Fig. 9.2.2(b), the weight or gravitational force W of a body is proportional to its mass m, where g is the gravitational acceleration. The value of g near the surface of the earth is 9.81 m/s^2, which is accurate enough for many engineering problems. In Fig. 9.2.2(c), the force of the spring is proportional to its elongation or displacement x, where k is the spring coefficient.

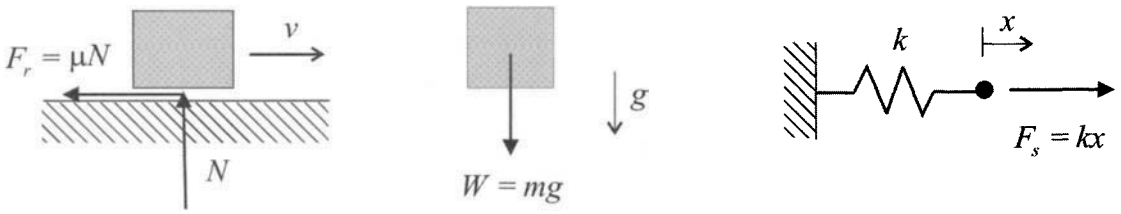
Moment The moment of a force is a measure of the tendency of the force to rotate a body. The moment $\boldsymbol{M}_O$ of force $\boldsymbol{F}$ about a reference point O is defined by the vector cross-product

$$\boldsymbol{M}_O = \boldsymbol{r} \times \boldsymbol{F} \tag{9.3}$$



(a) (b)

FIGURE 9.2.1 Sum $\boldsymbol{F} = \boldsymbol{F}_1 + \boldsymbol{F}_2$.



(a) Friction (b) Gravity (c) Spring force

FIGURE 9.2.2 Three examples of forces.

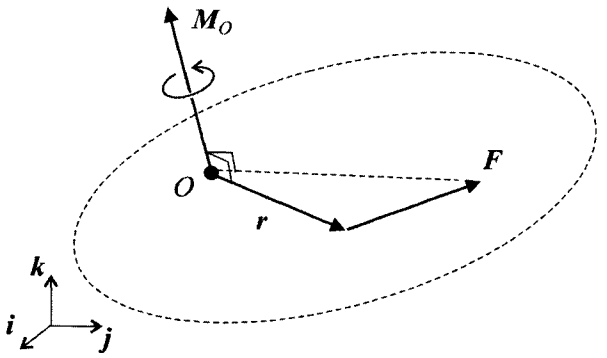


FIGURE 9.2.3 The moment of a force about point O.

where $\boldsymbol{r}$ is a position vector that runs from the reference point to the point of application of $\boldsymbol{F}$; see Fig. 9.2.3. The direction of the moment is determined by the right-hand rule, as shown in the figure. The moment in Eq. (9.3) can be expressed in the determinant form

$$\boldsymbol{M}_O = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ r_x & r_y & r_z \\ F_x & F_y & F_z \end{vmatrix} = (r_y F_z - r_z F_y)\boldsymbol{i} + (r_z F_x - r_x F_z)\boldsymbol{j} + (r_x F_y - r_y F_x)\boldsymbol{k} \tag{9.4}$$

where the vectors $\boldsymbol{r}$ and $\boldsymbol{F}$ are expressed by their Cartesian components

$$\boldsymbol{r} = r_x\boldsymbol{i} + r_y\boldsymbol{j} + r_z\boldsymbol{k}, \quad \boldsymbol{F} = F_x\boldsymbol{i} + F_y\boldsymbol{j} + F_z\boldsymbol{k} \tag{9.5}$$

with $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ being the unit vectors.

Resultant Force and Moment Consider a system of forces $\boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_3, \ldots$. Let the lth force and its point of application be

$$\boldsymbol{F}_l = F_{l,x}\boldsymbol{i} + F_{l,y}\boldsymbol{j} + F_{l,z}\boldsymbol{k}, \quad \boldsymbol{r}_l = r_{l,x}\boldsymbol{i} + r_{l,y}\boldsymbol{j} + r_{l,z}\boldsymbol{k} \tag{9.6}$$

The sum or resultant of the forces can be computed by

$$\boldsymbol{F} = \sum_l \boldsymbol{F}_l = \sum_l F_{l,x}\boldsymbol{i} + \sum_l F_{l,y}\boldsymbol{j} + \sum_l F_{l,y}\boldsymbol{k} \tag{9.7}$$

and the resultant of the moments about a point O of the forces is given by

$$\boldsymbol{M}_O = \sum_l (r_{l,y}F_{l,z} - r_{l,z}F_{l,y})\boldsymbol{i} + \sum_l (r_{l,z}F_{l,x} - r_{l,x}F_{l,z})\boldsymbol{j} + \sum_l (r_{l,x}F_{l,y} - r_{l,y}F_{l,x})\boldsymbol{k} \tag{9.8}$$

The Toolbox of this chapter provides a function `resultant` for computing resultant force and moment; see Window 2.1.

Window 2.1. Function resultant

MATLAB Function: **resultant**

Purpose:
To compute the resultants of a system of forces.

Synopsis:

```
resultant(F, R)
resultant(F)
[F_sum, M_sum] = resultant(F, R)
```

Description:
`resultant(F, R)` computes the resultant force and moment of a system of N forces described by matrices `F` and `R`, which have the form

$$\mathtt{F} = \begin{bmatrix} F_{1,x} & F_{1,y} & F_{1,z} \\ F_{2,x} & F_{2,y} & F_{2,z} \\ \vdots & \vdots & \vdots \\ F_{N,x} & F_{N,y} & F_{N,z} \end{bmatrix}, \qquad \mathtt{R} = \begin{bmatrix} r_{1,x} & r_{1,y} & r_{1,z} \\ r_{2,x} & r_{2,y} & r_{2,z} \\ \vdots & \vdots & \vdots \\ r_{N,x} & r_{N,y} & r_{N,z} \end{bmatrix}$$

Here, the lth row of `F` and that of `R` contain the components of the lth force and its point of application with respect to a reference point O, as defined in Eq. (9.6). If the problem is of two dimensions, namely, all the forces lie in the Oxy plane, only the first two columns of `F` and `R` need to be specified.

`resultant(F)` only computes the resultant force.

`[F_sum, M_sum] = resultant(F, R)` returns the resultant force in a vector `F_sum`, and the resultant moment in a vector `M_sum`.

EXAMPLE 2.1

In Fig. 9.2.4, four forces are applied to a body. Let O be the reference point. The matrices `F` and `R` in Window 2.1 are

$$\mathtt{F} = \begin{bmatrix} 0 & 70 \\ -110 \cdot \cos 30^\circ & 110 \cdot \sin 30^\circ \\ 60 \cdot \cos 45^\circ & 60 \cdot \sin 45^\circ \\ -50 & 0 \end{bmatrix}, \qquad \mathtt{R} = \begin{bmatrix} -1 & 2 \\ 2 & 4 \\ 5 & 5 \\ 5 & 1 \end{bmatrix}$$

Note that `F` and `R` are of two columns as the problem is two-dimensional. By the MATLAB commands

```
>> F = [0 70; -110*cos(pi/6) 110*sin(pi/6); 60*cos(pi/4)
   60*sin(pi/4); -50 0];
>> R = [-1 2; 2 4; 5 5; 5 1];
>> resultant(F,R)
```

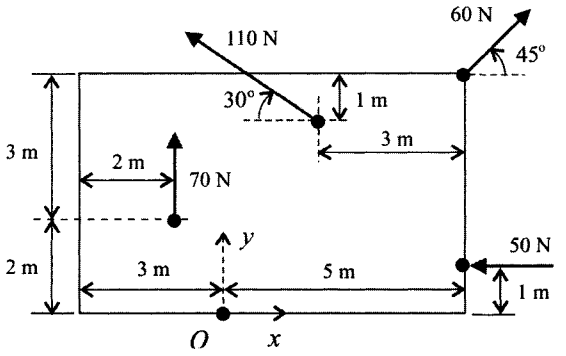


FIGURE 9.2.4

the resultants of the forces are obtained as follows

```
Resultant Force:
   F = -102.8364*i + (167.4264)*j + (0)*k
   Magnitude |F| = 196.4864

Resultant Moment:
   M = 0*i + (0)*j + (471.0512)*k
   Magnitude |M| = 471.0512
```

9.2.2 Kinematics

When a particle moves along a path in a reference frame, its position is described by a position vector $\boldsymbol{r}(t)$, which is a function of time; see Fig. 9.2.5. The velocity and acceleration of the particle are the first and second derivatives of $\boldsymbol{r}(t)$

$$\begin{aligned} \text{velocity} \quad & \boldsymbol{v}(t) = \frac{d\boldsymbol{r}(t)}{dt} = \dot{\boldsymbol{r}}(t) \\ \text{acceleration} \quad & \boldsymbol{a}(t) = \frac{d^2\boldsymbol{r}(t)}{dt^2} = \frac{d\boldsymbol{v}(t)}{dt} = \ddot{\boldsymbol{r}}(t) \end{aligned} \tag{9.9}$$

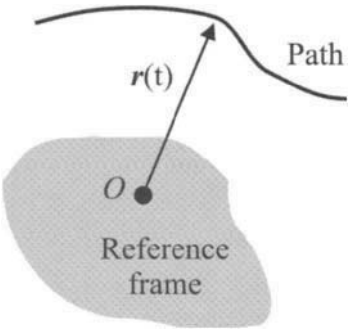


FIGURE 9.2.5 The path of a moving particle.

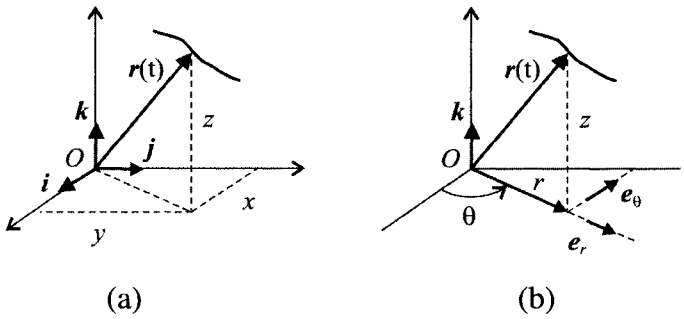


FIGURE 9.2.6 Two coordinate systems: (a) Cartesian coordinates; (b) cylindrical coordinates.

The motion (position, velocity, and acceleration) of a particle can be described in different coordinates. Two commonly used coordinate systems are shown in Fig. 9.2.6.

Rectangular Cartesian Coordinates As shown in Fig. 9.2.6(a), position, velocity, and acceleration vectors can be expressed by

$$\begin{aligned} \boldsymbol{r}(t) &= x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k} \\ \boldsymbol{v}(t) &= \dot{x}\boldsymbol{i} + \dot{y}\boldsymbol{j} + \dot{z}\boldsymbol{k} \\ \boldsymbol{a}(t) &= \ddot{x}\boldsymbol{i} + \ddot{y}\boldsymbol{j} + \ddot{z}\boldsymbol{k} \end{aligned} \tag{9.10}$$

For plane motion, say in the plane Oxy, Eqs. (9.10) are reduced to

$$\begin{aligned} \boldsymbol{r}(t) &= x\boldsymbol{i} + y\boldsymbol{j} \\ \boldsymbol{v}(t) &= \dot{x}\boldsymbol{i} + \dot{y}\boldsymbol{j} \\ \boldsymbol{a}(t) &= \ddot{x}\boldsymbol{i} + \ddot{y}\boldsymbol{j} \end{aligned} \tag{9.11}$$

Cylindrical Coordinates By Fig. 9.2.6(b), position, velocity, and acceleration vectors are of the form

$$\begin{aligned} \boldsymbol{r}(t) &= r\boldsymbol{e}_r + z\boldsymbol{k} \\ \boldsymbol{v}(t) &= \dot{r}\boldsymbol{e}_r + r\dot{\theta}\boldsymbol{e}_\theta + \dot{z}\boldsymbol{k} \\ \boldsymbol{a}(t) &= \left(\ddot{r} - r\dot{\theta}^2\right)\boldsymbol{e}_r + \left(r\ddot{\theta} + 2\dot{r}\dot{\theta}\right)\boldsymbol{e}_\theta + \ddot{z}\boldsymbol{k} \end{aligned} \tag{9.12}$$

where $\boldsymbol{e}_r$ and $\boldsymbol{e}_\theta$ are unit vectors whose directions change with angle θ. Equations (9.12) are derived based on the time derivatives of $\boldsymbol{e}_r$ and $\boldsymbol{e}_\theta$

$$\dot{\boldsymbol{e}}_r = \dot{\theta}\boldsymbol{e}_\theta, \qquad \dot{\boldsymbol{e}}_\theta = -\dot{\theta}\dot{\boldsymbol{e}}_r \tag{9.13}$$

For plane motion, the cylindrical coordinates are reduced to the polar coordinates (r, θ), and Eqs. (9.12) become

$$\begin{aligned} \boldsymbol{r}(t) &= r\boldsymbol{e}_r \\ \boldsymbol{v}(t) &= \dot{r}\boldsymbol{e}_r + r\dot{\theta}\boldsymbol{e}_\theta \\ \boldsymbol{a}(t) &= \left(\ddot{r} - r\dot{\theta}^2\right)\boldsymbol{e}_r + \left(r\ddot{\theta} + 2\dot{r}\dot{\theta}\right)\boldsymbol{e}_\theta \end{aligned} \tag{9.14}$$

9.2.3 Kinetics of Single Particle

Equations of Motion The motion of a particle is governed by Eq. (9.2). The equations of motion of a particle in the rectangular Cartesian coordinates are

$$\begin{aligned} ma_x &= \sum F_x \\ ma_y &= \sum F_y \\ ma_z &= \sum F_z \end{aligned} \tag{9.15}$$

where the acceleration components by Eq. (9.10) are $a_x = \ddot{x}$, $a_y = \ddot{y}$, $a_z = \ddot{z}$. In the cylindrical coordinates, the equations of motion are

$$\begin{aligned} ma_r &= \sum F_r \\ ma_\theta &= \sum F_\theta \\ ma_z &= \sum F_z \end{aligned} \tag{9.16}$$

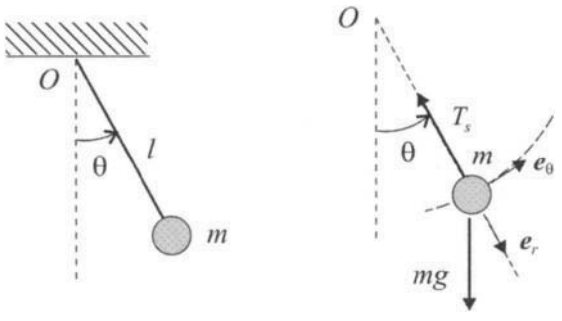
where the acceleration components are given by Eq. (9.12). Equations (9.15) and (9.16) are also applicable to particles in plane motion, for which the spatial coordinate z is not needed.

EXAMPLE 2.2

A simple pendulum (a lumped mass suspended by an unstretchable string) is in plane motion; see Fig. 9.2.7(a). The mass is subject to its weight mg and the string tension T_s, as shown in the free-body diagram in Fig. 9.2.7(b). By using the polar coordinates and Eqs. (9.16), the equations of motion of the simple pendulum are obtained as

$$\begin{aligned} ma_r &= -ml\dot{\theta}^2 = mg\cos\theta - T_s \\ ma_\theta &= ml\ddot{\theta} = -mg\sin\theta \end{aligned}$$

Here, zero derivatives of the string length l have been used.



(a) (b)

FIGURE 9.2.7

Work and Energy Dot-product Eq. (9.2) by $d\boldsymbol{r}$ and integrate the resulting equation between positions $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ on a path of the particle, to obtain

$$\int_{\boldsymbol{r}_1}^{\boldsymbol{r}_2} \boldsymbol{F} \cdot d\boldsymbol{r} = \frac{1}{2} \int_{t_1}^{t_2} m \frac{d}{dt}(\boldsymbol{v} \cdot \boldsymbol{v})\, dt = \frac{1}{2} m v^2 \Big|_{v_1}^{v_2} \tag{9.17}$$

where v is the magnitude of the velocity, i.e., $v = |\boldsymbol{v}|$. Define the kinetic energy of the particle as

$$T(t) = \frac{1}{2} m \boldsymbol{v} \cdot \boldsymbol{v} = \frac{1}{2} m v^2 \tag{9.18}$$

Define the work done by force $\boldsymbol{F}$ on a path from $\boldsymbol{r}_1$ to $\boldsymbol{r}_2$ as

$$W_{12} = \int_{\boldsymbol{r}_1}^{\boldsymbol{r}_2} \boldsymbol{F} \cdot d\boldsymbol{r} \tag{9.19}$$

By the above definitions, the integral in Eq. (9.17) is stated as the following *principle of work and energy*

$$W_{12} = T_2 - T_1 = \Delta T \tag{9.20}$$

where $T_1 = T(t_1)$ and $T_2 = T(t_2)$. Equation (9.20) shows that the work done by the force applied to the particle is equal to the change of its kinetic energy.

Potential Energy If the work W_{12} in Eq. (9.19) is independent of any path chosen between $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, force $\boldsymbol{F}$ is said to be *conservative*. A conservative force can be expressed as

$$\boldsymbol{F}(\boldsymbol{r}) = -\nabla V(\boldsymbol{r}) \tag{9.21}$$

where ∇ is the gradient operator (see Appendix A.8), and $V(\boldsymbol{r})$, a scalar function of position vector $\boldsymbol{r}$, is called *potential energy*. For example, the gravitational force in Fig. 9.2.2(b) and the spring force in Fig. 9.2.2(c) are both conservative, with the following potential functions:

$$\text{Gravity :} \qquad V = mgy$$

$$\text{Spring force :} \qquad V = \frac{1}{2} k x^2$$

where y is a spatial coordinate along an upward axis, and x is the spring elongation. On the other hand, the friction force in Fig. 9.2.2(a) is not conservative because its work is path-dependent.

According to Eqs. (9.19) and (9.21),

$$W_{12} = -\int_{\boldsymbol{r}_1}^{\boldsymbol{r}_2} \nabla V(\boldsymbol{r}) \cdot d\boldsymbol{r} = -(V_2 - V_1) = -\Delta V \tag{9.22}$$

in which $V_1 = V(\boldsymbol{r}_1)$ and $V_1 = V(\boldsymbol{r}_2)$. The above equation indicates that the work done by a conservative force is related to the change of its potential V. Let the resultant of the forces acting on the particle be the sum of a conservative force $\boldsymbol{F}^c$ and a nonconservative force $\boldsymbol{F}^{nc}$. By Eq. (9.22), the principle of work and energy given in Eq. (9.20) is rewritten as

$$W_{12}^{nc} = (T_2 + V_2) - (T_1 + V_1) = \Delta(T + V) \tag{9.23}$$

where W_{12}^{nc} is the work done by $\boldsymbol{F}^{nc}$, and V is the potential energy of the particle due to $\boldsymbol{F}^{c}$. The sum of kinetic energy and potential energy, $T + V$, is called the *mechanical energy* of the particle.

Conservation of Energy If all forces acting on the particle are conservative, $\boldsymbol{F}^{nc} = 0$. In this case, the mechanical energy of the particle is conserved:

$$T_2 + V_2 = T_1 + V_1 \tag{9.24}$$

A system (particle or body) whose mechanical energy is conserved is called *conservative system.*

EXAMPLE 2.3

The kinetic energy and potential energy of the simple pendulum in Fig. 9.2.7(a) are

$$T = \frac{1}{2}ml^2\dot{\theta}^2, \quad V = mgl(1 - \cos\theta).$$

The pendulum is conservative; that is, its total mechanical energy $T + V$ is constant.

Impulse and Momentum The principle of work and energy given by Eq. (9.20) is a spatial integral representation of Newton's second law. The temporal integration of Newton's second law, Eq. (9.2), leads to the following *principle of impulse and momentum*

$$\boldsymbol{I}_{12} = \boldsymbol{L}_2 - \boldsymbol{L}_1 \tag{9.25}$$

where $\boldsymbol{I}_{12} = \int_{t_1}^{t_2} \boldsymbol{F}dt$ is the impulse of the force applied to the particle, and $\boldsymbol{L} = m\boldsymbol{v}$ is the linear momentum of the particle.

Angular Momentum The angular momentum (moment of momentum) of a particle with respect to a reference point O is defined by the cross-product

$$\boldsymbol{H}_O = \boldsymbol{r} \times m\boldsymbol{v} \tag{9.26}$$

where $\boldsymbol{r}$ is the position vector from point O; see Fig. 9.2.8. Cross-product Eq. (9.2) by $\boldsymbol{r}$ and make use of Eq. (9.26), to obtain

$$\dot{\boldsymbol{H}}_O = \boldsymbol{r} \times \boldsymbol{F} = \boldsymbol{M}_O \tag{9.27}$$

Thus, the rate of change of the angular momentum equals the resultant moment.

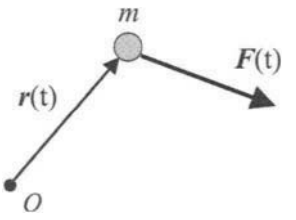


FIGURE 9.2.8

EXAMPLE 2.4

The equations of motion of the simple pendulum in Fig. 9.2.7(a) can be derived through angular momentum. To this end, derive the following quantities

$$\boldsymbol{H}_O = ml^2\dot{\theta}\cdot z, \quad \boldsymbol{M}_O = -mgl\sin\theta\cdot z$$

where $z = \boldsymbol{e}_r \times \boldsymbol{e}_\theta$. It follows from Eq. (9.26) that

$$ml^2\ddot{\theta} = -mgl\sin\theta$$

which is the same as the second equation of motion obtained in Example 2.2.

9.2.4 Particle Motion via Numerical Integration

Two basic problems in particle dynamics are (a) given motion, find external forces; and (b) given external forces, determine the motion of the body in consideration. In this section, problem (b) in Cartesian coordinates is solved by numerical integration. To this end, Eq. (9.15) is rewritten as

$$\ddot{x} = \frac{1}{m}\phi_1(t,\ \boldsymbol{u},\ \boldsymbol{v}), \quad \ddot{y} = \frac{1}{m}\phi_2(t,\ \boldsymbol{u},\ \boldsymbol{v}), \quad \ddot{z} = \frac{1}{m}\phi_3(t,\ \boldsymbol{u},\ \boldsymbol{v}) \tag{9.28}$$

where ϕ_j are the components of the resultant force

$$\phi_1 = \sum F_x, \quad \phi_2 = \sum F_y, \quad \phi_3 = \sum F_z \tag{9.29}$$

and $\boldsymbol{u}$ and $\boldsymbol{v}$ are displacement and velocity vectors

$$\boldsymbol{u}(t) = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \quad \boldsymbol{v}(t) = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} \tag{9.30}$$

Let the initial conditions of the particle be

$$\boldsymbol{u}(0) = \boldsymbol{u}_0, \quad \boldsymbol{v}(0) = \boldsymbol{v}_0 \tag{9.31}$$

The problem is that given external loads ϕ_j and the initial conditions (9.31), solve Eq. (9.28) for the response of the particle, $\boldsymbol{u}$ and $\boldsymbol{v}$.

Introduce the vector of state variables

$$\eta(t) = \begin{pmatrix} \boldsymbol{u}(t) \\ \boldsymbol{v}(t) \end{pmatrix} \tag{9.32}$$

and the vector of forcing functions

$$\boldsymbol{\phi} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{pmatrix} \tag{9.33}$$

Equations (9.28) and (9.31) are cast into an equivalent state-space form

$$\dot{\eta}(t) = \Phi(t, \eta(t)); \quad \eta(0) = \eta_0 \tag{9.34}$$

with

$$\Phi(t, \eta) = \begin{pmatrix} v \\ \frac{1}{m}\phi \end{pmatrix}, \quad \eta_0 = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \tag{9.35}$$

The state equation (9.34) is solved by a fixed-step Runge-Kutta method of order four:

$$\eta_{k+1} = \eta_k + \frac{h}{6}\left(f_1 + 2f_2 + 2f_3 + f_4\right), \quad k = 0, 1, 2, \ldots. \tag{9.36}$$

where $\eta_k = \eta(t_k)$, $t_k = kh$, h is the step size, and

$$\begin{aligned} f_1 &= \Phi(t_k, \eta_k) \\ f_2 &= \Phi\left(t_k + \frac{h}{2}, \eta_k + \frac{h}{2}f_1\right) \\ f_3 &= \Phi\left(t_k + \frac{h}{2}, \eta_k + \frac{h}{2}f_2\right) \\ f_4 &= \Phi\left(t_k + h, \eta_k + hf_3\right) \end{aligned} \tag{9.37}$$

In advanced numerical integration algorithms, the step size h is adaptively adjusted during the computation. For instance, the software MATLAB has Runge-Kutta solvers like `ode45` with automatic step-size adjustment capability.

The above formulation is for motion in three dimensions. For a particle in plane motion, simply use the following quantities

$$u(t) = \begin{pmatrix} x \\ y \end{pmatrix}, \quad v(t) = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix}, \quad \phi = \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} \tag{9.38}$$

The state equation (9.34) and numerical algorithm (9.36) remain the same.

The Toolbox of this chapter has functions `part2d` and `part3d` for solution of Eq. (9.34) by the numerical integration algorithm given in Eqs. (9.36); see Window 2.2. For these functions, the step size of the numerical integration is controlled by

$$h = \frac{\texttt{tf}}{\texttt{npts} - 1} \tag{9.39}$$

The user may try several step sizes to compare the accuracy and convergence of numerical results.

In addition, the Toolbox has a function `partmotion` for animation of particle motion in two or three dimensions; see Window 2.3.

Window 2.2. Functions part2d and part3d

MATLAB Functions: **part2d, part3d**

Purpose:
To plot the dynamic response of a particle by solving the state equation (9.34) through use of the Runge-Kutta algorithm given in Eq. (9.36).

Synopsis:

```
part2d(m, 'ForceFn', eta0)
part2d(m, 'ForceFn', eta0, tf, npts)
[t, eta, KE, F] = part2d(m, 'ForceFn', eta0, tf, npts)
```

Description:
`part2d(m, 'ForceFn', eta0)` plots the trajectory of a particle in a two-dimensional space, the time history of the displacements and kinetic energy of the particle, and the time history of the external forces acting on the particle. Here `m` is the mass of the particle, `ForceFn` is the name of a user-provided M-file function that describes the external forces ϕ_j in Eq. (9.29), and `eta0` is the vector of initial displacements and velocities given in Eq. (9.35). The user-provided function follows the format

```
f = ForceFn(t, eta)
```

where `t` is the time, `eta` is the state vector defined in Eq. (9.32), and `f` is the vector of forcing functions ϕ_1 and ϕ_2. If the particle has zero initial disturbances, simply call `part2d(m, 'ForceFn')`. If no external force acts on the particle, use `part2d(m, [ ], eta0)`.

`part2d(m, 'ForceFn', eta0, tf, npts)` plots the above-described particle response with specified parameters `tf` and `npts`, where `tf` is total simulation time and `npts` is the number of time points in the region $0 \leq t \leq$ `tf`. By default, `tf` = 10 and `npts` = 1,001. If there is no external force, '`ForceFn`' is replaced by 0 or [ ]. If the particle has zero initial disturbances, `eta0` is replaced by [ ].

`[t, eta, KE, F] = part2d(m, 'ForceFn', eta0, tf, npts)` returns the times of simulation in a vector `t`, the state vector in a four-row matrix `eta`, the kinetic energy of the particle in a vector `KE`, and the external forces (ϕ_1 and ϕ_2) in a two-row matrix `F`. The first two rows of `eta` contain the data of the displacements `x` and `y` of the particle, while the third and fourth rows the velocities $\dot{x}$ and $\dot{y}$. With these output variables, the motion of the particle can be plotted by the MATLAB function `plot`. For instance, `plot(t, eta (1,:), t, eta (2,:))` plots the trajectory of the particle in the 2-D space; and `plot(t, F(2,:))` plots the time history of the force component $\phi_2 = \sum F_y$.

The utility of function `part3d` is similar to that of `part2d`, except that the problem is solved in a 3-D space, with three coordinates (x, y, and z) and three force components (ϕ_1, ϕ_2, and ϕ_3).

Window 2.3. Function partmotion

MATLAB Function: **partmotion**

Purpose:
To animate the dynamic response of a particle in two or three dimensions.

Synopsis:

```
partmotion(t, eta)
partmotion(t, eta, IS_control, n_frames, frame_bounds)
[F, t_frame, z_frame] = partmotion(t, eta, IS_control, n_frames,
frame_bounds)
```

Description:
`animotion2d(t, eta)` plays the animated dynamic response of the particle, where `t` and `eta` are a vector of times and a matrix about state vector, respectively. The `t` and `eta` are obtained by function `rigid2d` or function `rigid3d`, like `[t, eta] = rigid2d(m, Iz, 'ForceFn', eta0)`.

`partmotion(t, eta, IS_control, n_frames, frame_bounds)` plays the animated response of the particle, with the specified parameters `IS_control`, `n_frames`, and `frame_bounds`. Here, `IS_control` is a control parameter with the following options:

```
IS_control = 0 play frames continuously, without particle trajectory preview
IS_control = 1 play frames continuously, with a particle trajectory preview
IS_control = 2 play frames one by one, without particle trajectory preview
IS_control = 3 play frames one by one, with a particle trajectory preview
```

`n_frames` is the total number of frames in the time period spanned by vector `t`; `frame_bounds = [xmin xmax ymin ymax]` or `[xmin xmax ymin ymax zmin zmax]` sets up the bounds of frames. By default, `IS_control = 0`, `n_frames = 101`, and `frame_bounds` is automatically set up according to matrix `eta`. The `IS_control`, `n_frames`, `frame_bounds` can be replaced by `[ ]` as a placeholder for their default setup.

`[F, t_frame, z_frame] = partmotion(t, eta, IS_control, n_frames, frame_bounds)` returns a set of frames of the motion of the rigid body in a matrix `F`, which later on can be played by the MATLAB function `movie(F)`, the times for the frames in a vector `t_frame`, and the displacements of the particle at the frame times in a vector `z_frame`.

EXAMPLE 2.5

A rock of 0.1 kg is thrown at an elevation of 2 m with an initial velocity shown in Fig. 9.2.9. The aerodynamic drag acting on the rock is

$$F_D = 0.005v^2, \text{in newton}$$

where $v = \sqrt{\dot{x}^2 + \dot{y}^2}$ is the magnitude of the velocity. The equations of motion of the

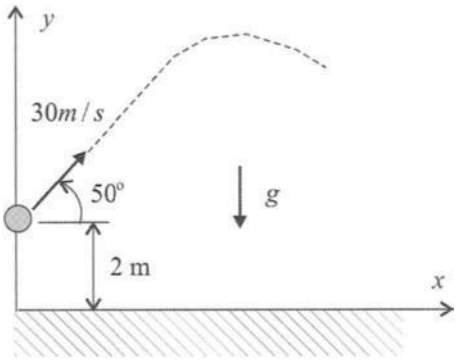


FIGURE 9.2.9

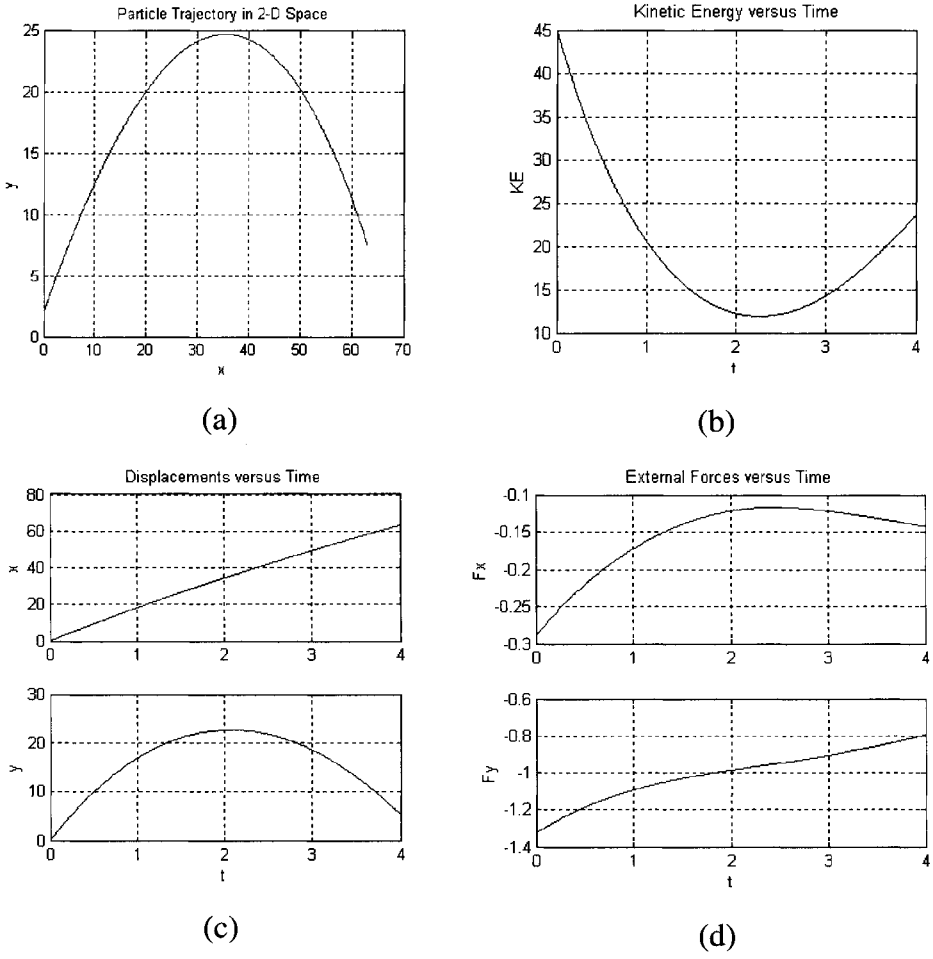


FIGURE 9.2.10

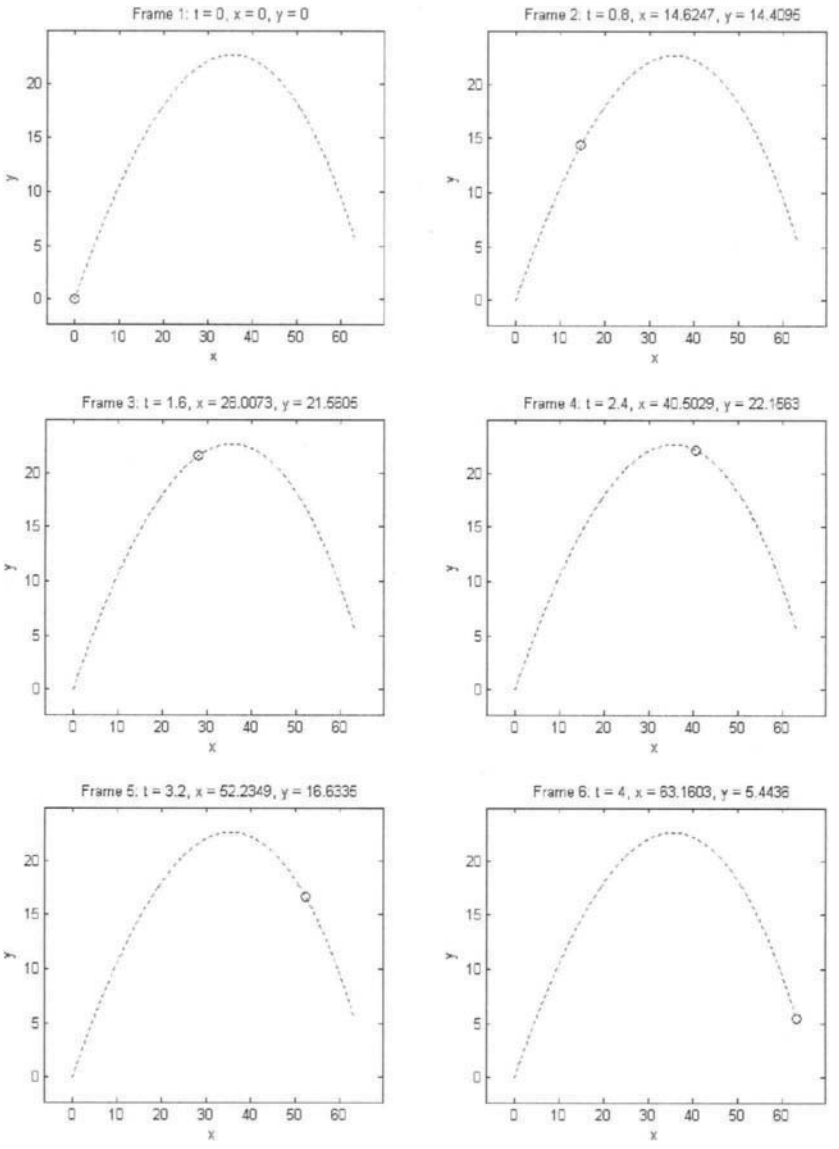


FIGURE 9.2.11

rock are

$$m\ddot{x} = -0.005\dot{x}v, \quad m\ddot{y} = -0.005\dot{y}v - mg$$

where $g = 9.81$ m/s is the gravitational acceleration. This is a two-dimensional problem. So, function `part2d` is used to determine the rock motion. To describe the external forces, an M-file function `force_exm25` is created as follows:

```
% M-file named force_exm25
function f = force_exm25(tt, z)
m = 0.1;
```

```
vx = z(3); vy = z(4);
v = sqrt(vx^2 + vy^2);
f = [-0.0005*vx*v; -0.0005*vy*v-9.81*m];
```

(See Appendix B.4 for how to generate an M-file.) By the MATLAB commands

```
>> eta0 = [0;  2;  30*cos(pi/180*50);  30*sin(pi/180*50)];
>> part2d(0.1, 'force_exm25', eta0, 4)
```

the trajectory, displacements, and kinetic energy of the rock and external forces are plotted in Fig. 9.2.10, for $0 \leq t \leq 4$ s. Also

```
>> [t, eta] = part2d(0.1, 'force_exm25', eta0, 4);
>> partmotion(t, eta, 3, 6)
```

plays the animated motion of the particle; see Fig. 9.2.11 for six frames of the animated motion.

EXAMPLE 2.6

In Fig. 9.2.12, a ball is suspended at O in a three-dimensional space by a *stretchable* string of initial length l_0. The string, when stretched, can be modeled as a spring of coefficient k. However, the string cannot sustain compression. Thus, the spring force applied to the ball by the string is

$$F_s = \begin{cases} k\,(l - l_0), & \text{for } l > l_0 \\ 0, & \text{for } l \leq l_0 \end{cases}$$

where $l = \sqrt{x^2 + y^2 + z^2}$ is the distance between the ball and the origin. The equations of motion of the ball in the 3-D space are

$$m\ddot{x} = -F_s \frac{x}{l}, \quad m\ddot{y} = -F_s \frac{y}{l}, \quad m\ddot{z} = -F_s \frac{z}{l} - mg$$

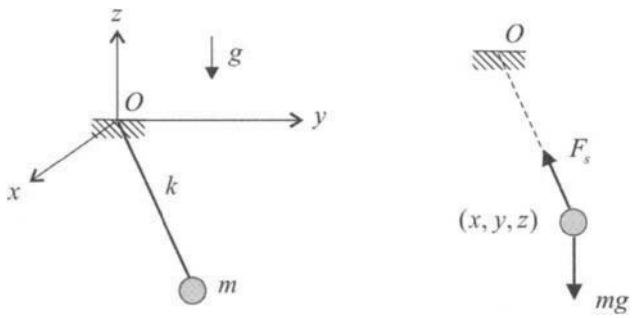


FIGURE 9.2.12

Let the parameters of the "pendulum" be $m = 10$ kg, $l_0 = 1$ m, and $k = 5{,}000$ N/m. Assume the initial conditions of the ball are

$$x(0) = 0.6 \text{ m}, \quad y(0) = 0, \quad z(0) = -0.8 \text{ m}$$
$$\dot{x}(0) = 0, \quad \dot{y}(0) = -2 \text{ m/s}, \quad \dot{z}(0) = 0$$

The external forces are described by the following M-file function

```
% M-file named force_exm26
function f = force_exm26(tt, a)
m = 10; L0 = 1; ks = 5000;
x = a(1); y = a(2); z = a(3);
Lb = sqrt(x^2 + y^2 + z^2);
if Lb > L0
   Fs = ks*(Lb-L0);
   f = [-Fs*x/Lb; -Fs*y/Lb; -Fs*z/Lb-m*9.81];
else
   f = [0; 0; -m*9.81];
end
```

```
>> part3d(10, 'force_exm26', [0.6 0 -0.8 0 -2 0]', 1.5)
```

plots the trajectory of the ball for $0 \leq t \leq 1.5$ s in Fig. 9.2.13, the kinetic energy of the ball in Fig. 9.2.14, and the external forces acting on the ball in Fig. 9.2.15.

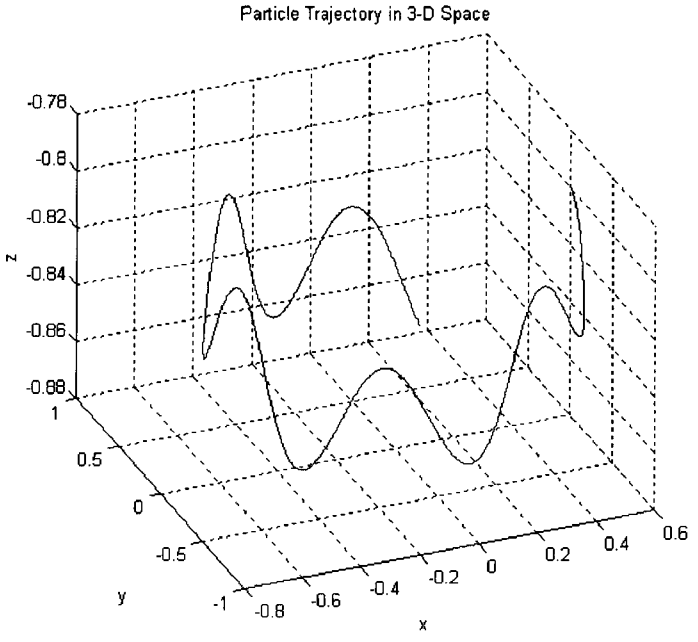


FIGURE 9.2.13

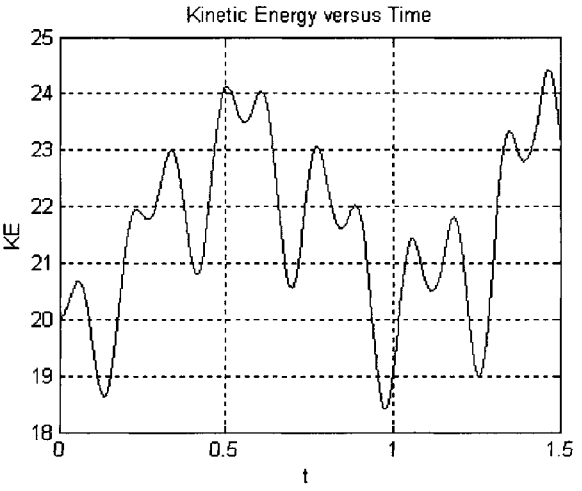


FIGURE 9.2.14

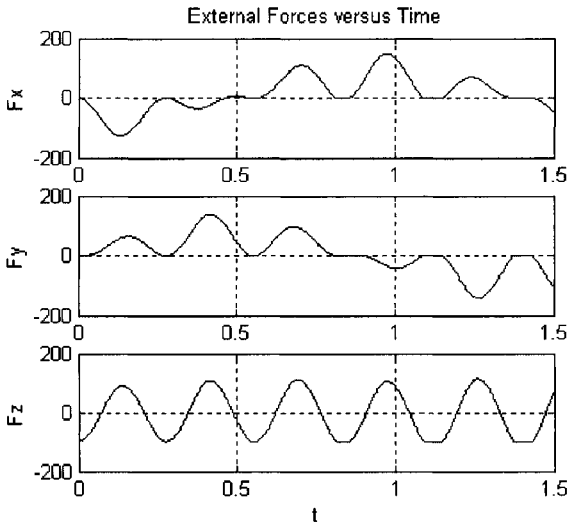


FIGURE 9.2.15

9.2.5 Systems of Particles

Consider a system of particles, as shown in Fig. 9.2.16, where $\boldsymbol{r}_i(t)$ is the position vector of the ith mass, $\boldsymbol{f}_{ij}$ is the force exerted to the ith particle by the jth particle, and $\boldsymbol{F}_i$ is the resultant of the forces exerted onto the ith particle by objects outside the system. By Newton's third law, $\boldsymbol{f}_{ij}$ and $\boldsymbol{f}_{ji}$ are action and reaction forces, that is, $\boldsymbol{f}_{ij} = -\boldsymbol{f}_{ji}$. It follows that

$$\sum_{i,j} \boldsymbol{f}_{ij} = 0, \quad \sum_{i,j} \boldsymbol{r}_i \times \boldsymbol{f}_{ij} = 0 \tag{9.40}$$

We shall call $\boldsymbol{f}_{ij}$ *internal* forces, and $\boldsymbol{F}_i$ *external* forces.

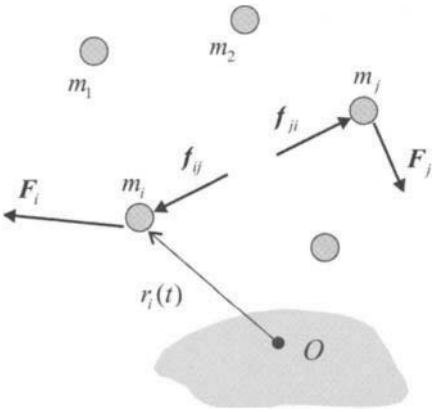


FIGURE 9.2.16 A system of particles.

Euler's Laws The equation of motion for ith particle in Fig. 9.2.16 is

$$m_i \boldsymbol{a}_i = \boldsymbol{F}_i + \sum_j \boldsymbol{f}_{ij} \tag{9.41}$$

Summation of the above equation over all particles, $\sum_i m_i \boldsymbol{a}_i = \sum_i \boldsymbol{F}_i + \sum_{i,j} \boldsymbol{f}_{ij}$, yields the following equation about the mass center of the particle system

$$\sum_i \boldsymbol{F}_i = \sum_i m_i \boldsymbol{a}_i \tag{9.42}$$

Equation (9.42) is the particle-system form of Euler's first law. Also, taking the cross produce of $\boldsymbol{r}_i$ with both sides of Eq. (9.41) and summing the resulting equation over i, leads to

$$\sum_i \boldsymbol{M}_{i,O} = \sum_i \boldsymbol{r}_i \times \boldsymbol{F}_i = \sum_i \boldsymbol{r}_i \times m_i \boldsymbol{a}_i \tag{9.43}$$

where $\boldsymbol{M}_{i,O} = \boldsymbol{r}_i \times \boldsymbol{F}_i$ is the moment about O of the resultant external force applied to the ith particle. Equation (9.43) is Euler's second law about particle systems.

Linear and Angular Momentum The linear momentum of a particle system is defined as

$$\boldsymbol{L}(t) = \sum_i m_i \boldsymbol{v}_i \tag{9.44}$$

By this definition, Eq. (9.42) is reduced to

$$\sum_i \boldsymbol{F}_i = \dot{\boldsymbol{L}} \tag{9.45}$$

Integration of the above equation gives the following principle of impulse and momentum for the particle system

$$\int_{t_1}^{t_2} \sum_i \boldsymbol{F}_i \, dt = \boldsymbol{L}(t_2) - \boldsymbol{L}(t_1) \tag{9.46}$$

Likewise, the angular momentum about reference point O of a particle system is defined as

$$\boldsymbol{H}_O(t) = \sum_i (\boldsymbol{r}_i \times m_i \boldsymbol{v}_i) \tag{9.47}$$

where $\boldsymbol{v}_i$ is the velocity of the ith particle. By Eq. (9.47), Eq. (9.43) is reduced to

$$\sum_i \boldsymbol{M}_{i,O} = \dot{\boldsymbol{H}}_O \tag{9.48}$$

Integration of the above equation gives another principle of impulse and momentum

$$\int_{t_1}^{t_2} \sum_i \boldsymbol{M}_{i,O} \, dt = \boldsymbol{H}_O(t_2) - \boldsymbol{H}_O(t_1) \tag{9.49}$$

If no external force acts on the particles of the system, $\dot{\boldsymbol{L}} = 0$ and $\dot{\boldsymbol{H}}_O = 0$, which means that the momentum of the system is conserved:

$$\boldsymbol{L}(t) = \text{constant}, \qquad \boldsymbol{H}_O(t) = \text{constant} \tag{9.50}$$

Because Eqs. (9.50) are of vector form, conservation of momentum may be possible along one direction. For instance, if $\sum_i F_{i,x} = 0$ but $\sum_i F_{i,y}$ and $\sum_i F_{i,z}$ are nonzero, the linear momentum in the x-direction is conserved: $L_x(t) = \text{constant}$.

Mass Center The mass center of a particle system is defined as

$$\boldsymbol{r}_G = \frac{1}{m} \sum_i m_i \boldsymbol{r}_i \tag{9.51}$$

where $m = \sum_i m_i$ is the total mass of the system. The velocity and acceleration of the mass center of the system are then

$$\begin{aligned} \boldsymbol{v}_G &= \frac{d\boldsymbol{r}_G}{dt} = \frac{1}{m} \sum_i m_i \boldsymbol{v}_i \\ \boldsymbol{a}_G &= \frac{d^2\boldsymbol{r}_G}{dt^2} = \frac{1}{m} \sum_i m_i \boldsymbol{a}_i \end{aligned} \tag{9.52}$$

where $\boldsymbol{v}_i$ and $\boldsymbol{a}_i$ are the velocity and acceleration of the ith particle, respectively.

The Toolbox of this chapter has a function `mscenter` for computing mass center for a given particle system; see Window 2.4.

Window 2.4. Function mscenter

MATLAB Function: **mscenter**

Purpose:
To compute the mass center and total mass of a system of particles.

Synopsis:

```
mscenter(Sys_Spec)
[rG, m_tot] = mscenter(Sys_Spec)
```

Description:
`mscenter(Sys_Spec)` computes the mass center and total mass of a system of N particles that is described by the matrix

$$\text{Sys\_Spec} = \begin{bmatrix} m_1 & x_1 & y_1 & z_1 \\ m_2 & x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots & \vdots \\ m_N & x_N & y_N & z_N \end{bmatrix}$$

The *j*th row of `Sys_Spec` contains the mass and Cartesian coordinates of the *j*th particle. For a two-dimensional problem, specification of the fourth column of `Sys_Spec` (i.e., the z-coordinates) is not needed.

`[rG, m_tot] = mscenter(Sys_Spec)` returns the position of mass center (in Cartesian coordinates) in a vector `rG`, and the total mass in `m_tot`.

EXAMPLE 2.7

For the particle system in Fig. 9.2.17, the matrix Sys_Spec in Window 2.3 is

$$\text{Sys\_Spec} = \begin{bmatrix} 25 & 3 & 1 \\ 15 & 1 & 2 \\ 10 & -3 & 3 \\ 30 & -3 & -1 \\ 20 & 0 & -2 \end{bmatrix}$$

By the MATLAB commands

```
>> Sys_Spec = [25 3 1; 15 1 2; 10 -3 3; 30 -3 -1; 20 0 -2];
>> mscenter(Sys_Spec)
```

the total mass and mass center of the system are obtained as follows

```
Total mass = 100 kg
Mass center:  xG = -0.3 m, yG = 0.15 m
```

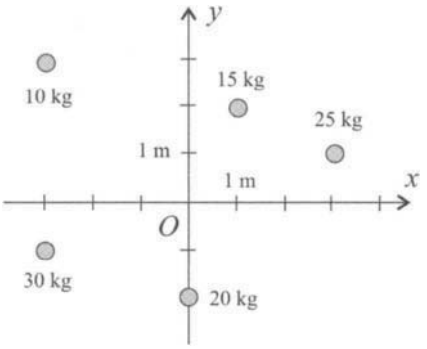


FIGURE 9.2.17

Motion of Mass Center Substituting Eq. (9.52) into Eq. (9.42) gives the equation of motion of the mass center of a particle system

$$\sum_i \boldsymbol{F}_i = m\boldsymbol{a}_G \tag{9.53}$$

The above equation states that the sum of the external forces is proportional to the acceleration of the mass center.

Consider the two frames shown in Fig. 9.2.18:

(i) a fixed frame $Oxyz$; and
(ii) a moving frame $Gx'y'z'$ that translates with respect to the fixed frame, with the origin being the mass center G of the particle system.

The position vectors of particle m_i in these frames are related by

$$\boldsymbol{r}_i = \boldsymbol{r}_G + \boldsymbol{\rho}_i \tag{9.54}$$

where $\boldsymbol{r}_G$ is given in Eq. (9.51). Substituting Eq. (9.54) into Eq. (9.43) yields

$$\sum_i \boldsymbol{M}_{i,G} = \dot{\boldsymbol{H}}_G \tag{9.55}$$

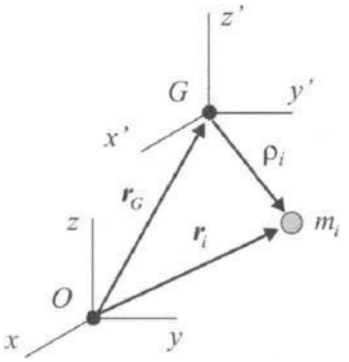


FIGURE 9.2.18 A fixed frame and a moving frame.

where $\boldsymbol{M}_{i,G} = \boldsymbol{\rho}_i \times \boldsymbol{F}_i$ is the moment about the mass center G of the external forces acting on the ith particle; $\boldsymbol{H}_G$ is the angular momentum about G, which is given by

$$\boldsymbol{H}_G = \sum_i \boldsymbol{\rho}_i \times m_i \boldsymbol{v}_i = \sum_i \boldsymbol{\rho}_i \times m_i \boldsymbol{v}_i' \tag{9.56}$$

with $\boldsymbol{v}_i' = \dot{\boldsymbol{\rho}}_i$ being the velocity of particle m_i measured in the moving frame.

Principle of Work and Energy For the particle system in Fig. 9.2.16, the principle of work and energy follows Eq. (9.20):

$$\sum_i W_i = \sum_i \Delta T_i \tag{9.57}$$

where $T_i = \frac{1}{2} m_i |\boldsymbol{v}_i|^2$ is the kinetic energy of the ith particle, and W_i is the work done by all the forces (external and internal) acting on the ith particle. Furthermore, through introduction of potential energy, Eq. (9.57) is reduced to

$$\sum_i W_i^{nc} = \sum_i \Delta T_i + \Delta V \tag{9.58}$$

where W_i^{nc} is the work done by nonconservative forces acting on the ith particle, and V is the potential energy of the system. The terms on the right-hand side of Eq. (9.58) represent the mechanical energy of the system. The mechanical energy is conserved if $\sum_i W_i^{nc} = 0$.

Impact The collision of two particles A and B along a straight line is shown in Fig. 9.2.19, where (a), (b), and (c) depict the status of the particles just before, during, and immediately after the collision. Assume that in the direction of the motion, there does not exist any external force during the collision. The linear momentum for the system composed of particles A and B is conserved:

$$m_A v_{A1} + m_B v_{B1} = m_A v_{A2} + m_B v_{B2} \tag{9.59}$$

Introduce the coefficient of restitution e

$$e = \frac{v_{B2} - v_{A2}}{v_{A1} - v_{B1}} \tag{9.60}$$

which is a measure of the capacity of the particles to bounce off each other after the collision. The value of e falls between 0 and 1, depending on the materials and geometry of the colliding bodies. When $e = 1$, no energy is lost during the collision. In this case, the impact is

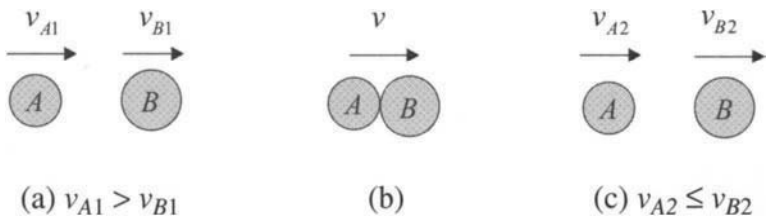


FIGURE 9.2.19 Rectilinear collision of two particles.

perfectly elastic. When $e = 0$, there is loss in kinetic energy during the collision, and the two particles stick together after the impact ($v_{A2} = v_{B2}$). In this case, the impact is perfectly plastic. For $0 < e < 1$, energy dissipation is involved during the collision, although the particles are separated after the impact.

EXAMPLE 2.8

In Fig. 9.2.20(a), a simple pendulum is at rest when a ball of mass M hits it in the horizontal direction. After the impact, the ball and pendulum stick together; see Fig. 9.2.20(b). If the maximum swing angle of the ball-pendulum system is θ_f, what is the velocity v of the ball just before the collision?

Solution. The ball and the pendulum are two parts of a particle system. Because the impact is plastic, $e = 0$ and

$$Mv + m \cdot 0 = (M + m)v_1$$

where v_1 is the velocity of the ball-pendulum system immediately after the collision. The mechanical energy of the ball-pendulum system in the slewing motion shown in Fig. 9.2.20(b) is obtained in the following cases:

$$\text{at } \theta = 0: \quad T_1 + V_1 = \frac{1}{2}(M + m)v_1^2 + 0$$

$$\text{at } \theta = \theta_f: \quad T_2 + V_2 = 0 + (M + m)gl(1 - \cos\theta_f)$$

Because the system is conservative, $T_1 + V_1 = T_2 + V_2$, which leads to

$$v_1^2 = 2gl(1 - \cos\theta_f)$$

From the above equations, the velocity v is expressed in θ_f by

$$v = \frac{M + m}{M} \cdot \sqrt{2gl(1 - \cos\theta_f)}.$$

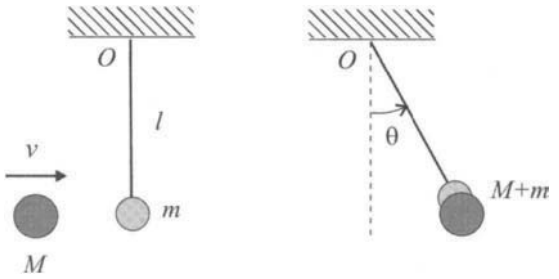


FIGURE 9.2.20

9.2.6 Central Force Motion

Central Force In Fig. 9.2.21(a), a particle is subject to a force that is always directed toward a fixed point O. Such a force is called *central force*, and O the center of force. The moment of

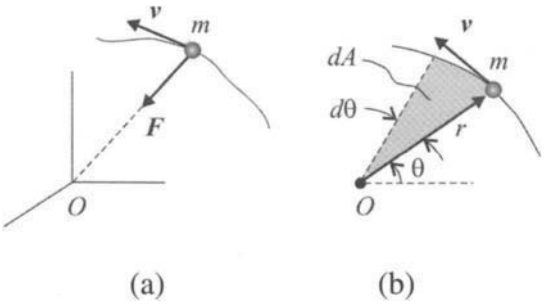


FIGURE 9.2.21 A particle subject to a central force.

a central force about the center is always zero. So, the angular momentum of a particle moving under a central force is constant, i.e.,

$$\boldsymbol{H}_O = \boldsymbol{r} \times m\boldsymbol{v} = \text{constant}$$

Accordingly, the particle shall move in a plane perpendicular to $\boldsymbol{H}_O$, and its angular momentum in such a plane as shown in Fig. 9.2.21(b) is

$$H_O = mr^2\dot{\theta} = \text{constant} \tag{9.61}$$

On the other hand, the area swept out by $\boldsymbol{r}$ during a time dt is $dA = \frac{1}{2}\boldsymbol{r} \cdot \boldsymbol{r}d\theta$; see Fig. 9.2.21(b). Thus, the rate of the area swept by $\boldsymbol{r}$, which is referred to as *areal velocity*, is

$$\frac{dA}{dt} = \frac{1}{2}r^2\dot{\theta} = \frac{H_O}{2m} \tag{9.62}$$

Equation (9.62) shows that the areal velocity of a particle under a central force is constant.

Newton's Law of Gravitation Consider two particles of mass m_1 and m_2 in Fig. 9.2.22. According to the law of gravitation, the two particles are attracted to each other along their connecting line by forces of magnitude

$$F = G\frac{m_1 m_2}{r^2} \tag{9.63}$$

where G is the universal gravitational constant, and is equal to $6.67 \times 10^{-11}\ m^3/(kg \cdot s^2)$.

As an example, consider a body of mass m on the earth with mass M and mean radius R. (The mass of the earth is 5.98×10^{24} kg; the mean radius of the earth is 6.37×10^6 m.)

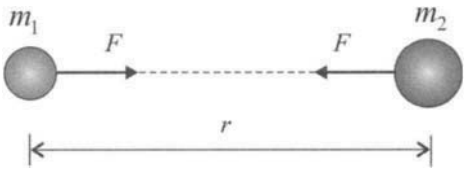


FIGURE 9.2.22

By Eq. (9.63), the attraction force applied to the body by the earth, or the weight of the body, is $W = mg$, where $g = GM/R^2 = 9.8\,m/s^2$ is the gravitational acceleration as shown in Fig. 9.2.2(b).

Equations of Motion The motion of a particle under a central force is governed by

$$\begin{aligned} ma_r &= m(\ddot{r} - r\dot{\theta}^2) = -F \\ ma_\theta &= m(r\ddot{\theta} + 2\dot{r}\dot{\theta}) = 0 \end{aligned} \tag{9.64}$$

where the polar coordinates in the plane in Fig. 9.2.21 have been used. Based on Eqs. (9.64), the trajectory of the particle is defined by the differential equation

$$\frac{d^2u}{d\theta^2} + u = \frac{F}{mh^2u^2} \tag{9.65}$$

where $u = \frac{1}{r}$, and

$$h = \frac{H_O}{m} = r^2\dot{\theta} = \text{constant} \tag{9.66}$$

Trajectory of a Space Vehicle in Free Flight Consider a space vehicle in free flight about the earth; see Fig. 9.2.23. Compared to the earth, the space vehicle is small enough to be viewed as a particle. The attraction force applied to the space vehicle by the earth is a central force, which, according to the law of gravitation, is given by

$$F = \frac{GMm}{r^2} = GMmu^2 \tag{9.67}$$

where M is the mass of the earth. Substitution of Eq. (9.67) into Eq. (9.65) gives

$$\frac{d^2u}{d\theta^2} + u = \frac{GM}{h^2} \tag{9.68}$$

The solution of Eq. (9.68) is of the form

$$u = \frac{1}{r} = \frac{GM}{h^2} + B\cos(\theta - \theta_0) \tag{9.69}$$

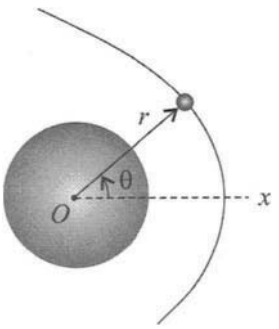


FIGURE 9.2.23 A space vehicle in free flight.

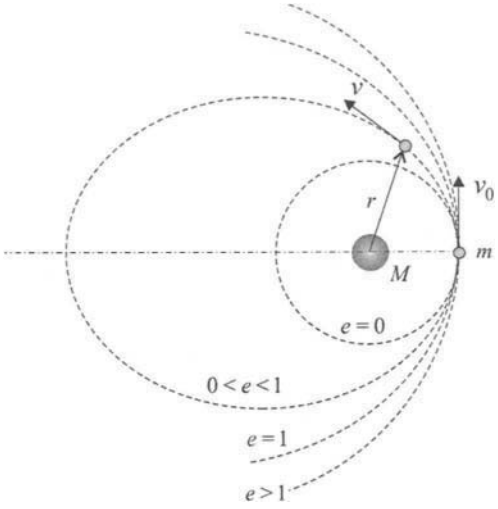


FIGURE 9.2.24 Four possible trajectories of a space vehicle.

which represents a conic section. Without loss of generality, set $\theta_0 = 0$, which means that the x-axis in Fig. 9.2.23 is the axis of symmetry of the conic section. Equation (9.69) thus is written as

$$\frac{1}{r} = \frac{1}{ep} + \frac{1}{p}\cos\theta \tag{9.70}$$

where $e = \frac{Bh^2}{GM}$ is the eccentricity of the conic section, and $p = \frac{1}{B}$ is the distance from the focus to the directrix. (Refer to Appendix A.7 for more detail on a conic section.)

There are four possible trajectories, depending on the value of the eccentricity e (Fig. 9.2.24):

(i) $e = 0$ or $B = 0$: the conic section is a circle;
(ii) $e < 1$ or $B > GM/h^2$: the conic section is an ellipse;
(iii) $e = 1$ or $B = GM/h^2$: the conic section is a parabola; and
(iv) $e > 1$ or $B > GM/h^2$: the conic section is a hyperbola.

Let the space vehicle start the free flight at $\theta = 0$, with initial distance r_0 from the earth center and initial velocity $v_0 = r_0\dot{\theta}_0$. Substitute these initial values into Eq. (9.69) (with $\theta_0 = 0$) to obtain

$$B = \frac{1}{r_0} - \frac{GM}{h^2} = \frac{1}{r_0} - \frac{GM}{r_0^2 v_0^2} \tag{9.71}$$

where $h = r_0^2 v_0$ by Eq. (9.66). Substituting Eq. (9.71) into the condition $B = GM/h^2$ of the parabolic trajectory yields the initial velocity $v_0 = \sqrt{2GM/r_0}$. This is the least initial velocity for which the space vehicle does not return to its starting point. Thus, the velocity is termed as *escape velocity*, and is written as

$$v_{\text{esc}} = \sqrt{\frac{2GM}{r_0}} \tag{9.72}$$

Following the above discussion, the conditions on the initial velocity for all the possible trajectories are given in the table below:

Circular Trajectory	*Elliptic Trajectory*	*Parabolic Trajectory*	*Hyperbolic Trajectory*
$v_0 = \sqrt{\frac{GM}{r_0}}$	$\sqrt{\frac{GM}{r_0}} < v_0 < \sqrt{\frac{2GM}{r_0}}$	$v_0 = \sqrt{\frac{2GM}{r_0}}$	$v_0 > \sqrt{\frac{2GM}{r_0}}$

The above discussion is also applicable to the free flight of spacecraft around other planets.

9.3 Dynamics of Rigid Bodies in Plane Motion

9.3.1 Plane Motion

A rigid body is an object in which the distance between any two points remains the same. By definition, a rigid body does not deform, or change shape. In this section, the two-dimensional motion of slablike rigid bodies is considered.

Translation Translation is a motion in which any straight line in a body keeps the same direction during the motion; see Fig. 9.3.1. In translation, all points of the body have the same velocity and the same acceleration, i.e.,

$$\boldsymbol{v}_A = \boldsymbol{v}_B = \boldsymbol{v}_C$$
$$\boldsymbol{a}_A = \boldsymbol{a}_B = \boldsymbol{a}_C$$

Rotation about a Point The rotation of a body is shown in Fig. 9.3.2, which is a motion about an axis that is perpendicular to plane Oxy and passes reference point O. The velocity of any point P is

$$\boldsymbol{v} = \boldsymbol{\omega} \times \boldsymbol{r} = \omega r \boldsymbol{k} \tag{9.73}$$

and the acceleration at P is

$$\boldsymbol{a} = \boldsymbol{a}_t + \boldsymbol{a}_n \tag{9.74a}$$

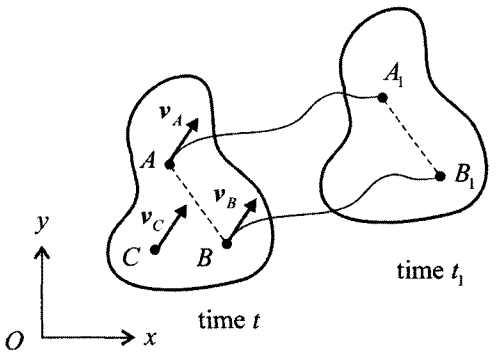


FIGURE 9.3.1 A rigid body in translation.

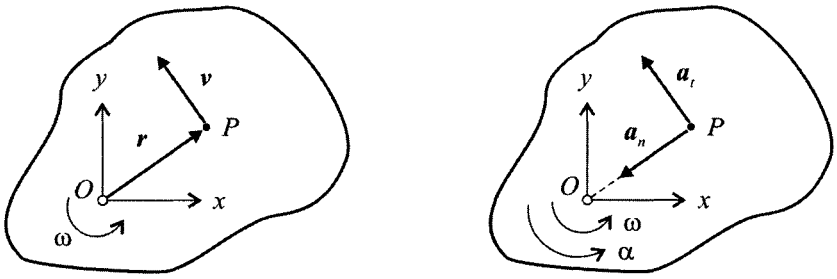


FIGURE 9.3.2 The rotation of a rigid body about a point.

with the tangential and normal acceleration components

$$\begin{aligned} \boldsymbol{a}_t &= \dot{\boldsymbol{\omega}} \times \boldsymbol{r} = \alpha \boldsymbol{k} \times \boldsymbol{r} \\ \boldsymbol{a}_n &= \boldsymbol{\omega} \times \dot{\boldsymbol{r}} = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r}) = -\omega^2 \boldsymbol{r} \end{aligned} \tag{9.74b}$$

Here $\boldsymbol{\omega}$ is the angular velocity, ω and α are the magnitudes of the angular velocity and acceleration, and $\boldsymbol{k}$ is a unit vector that is perpendicular to plane Oxy (determined by the right-hand rule). The magnitudes of the velocity and acceleration are given by

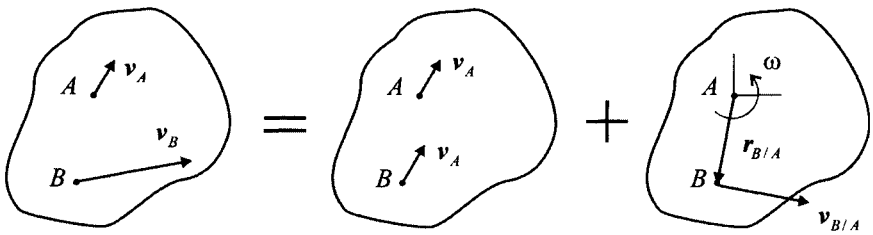
$$v = \omega r, \quad a_t = \alpha r, \quad a_n = -\omega^2 r \tag{9.75}$$

where $r = |\boldsymbol{r}|$. The angular velocity and acceleration can be expressed as

$$\omega = \dot{\theta}, \quad \alpha = \dot{\omega} = \ddot{\theta} \tag{9.76}$$

where θ is the rotational displacement of the body. The unit of θ is radians (rad), the unit of ω is rad/s, and the unit of α is rad/s^2.

General Plane Motion A general plane motion can be decomposed into a translation defined by the motion of any reference point and a simultaneous rotation about that point. For instance, let A be a reference point of the body in Fig. 9.3.3. The velocity $\boldsymbol{v}_B$ of point B is the sum of the



Plane motion Translation with A Rotation about A

FIGURE 9.3.3 General motion as a sum of translation and rotation.

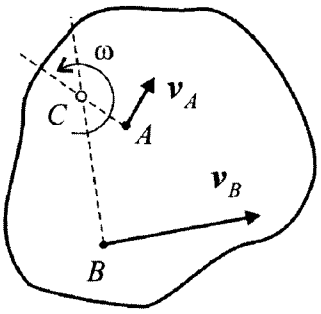


FIGURE 9.3.4 A rigid body with instantaneous center of rotation.

velocity v_A of the translation with A and the velocity $v_{B/A}$ relative to A:

$$v_B = v_A + v_{B/A} \tag{9.77}$$

The $v_{B/A}$ is a relative velocity associated with rotation about A

$$v_{B/A} = \dot{r}_{B/A} = \omega \times r_{B/A} = \omega k \times r_{B/A} \tag{9.78}$$

The angular velocity ω is independent of the reference point. In other words, any two points of a rigid body have the same angular velocity. The acceleration of point B is

$$a_B = a_A + a_{B/A} \tag{9.79}$$

where $a_{B/A}$ is a relative acceleration associated with rotation about A

$$a_{B/A} = \dot{v}_{B/A} = \alpha k \times r_{B/A} - \omega^2 r_{B/A} \tag{9.80}$$

Instantaneous Center of Rotation The instantaneous center of a rigid body is a point whose velocity is zero at the instant under consideration. The motion of the body at the instant can be viewed as a rotation about the instantaneous center. In Fig. 9.3.4, the instantaneous center C is determined by velocities at A and B. Since $v_C = 0$, the velocities at A and B by Eq. (9.77) are

$$v_A = \omega k \times r_{A/C}, \quad v_B = \omega k \times r_{B/C}$$

The instantaneous center may or may not be a point of the rigid body.

EXAMPLE 3.1

In Fig. 9.3.5, a disk is rolling on a horizontal surface with rotation speed ω. Assume that there is no slipping at B, such that the velocity at the point is zero. If the disk center C is taken as a reference point, the motion of the disk is the sum of the translation with C and rotation about C. The velocity at B is

$$v_B = v_C - R\omega$$

Because $v_B = 0$,

$$v_C = R\omega.$$

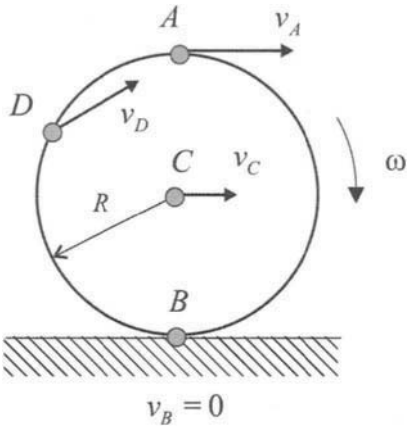


FIGURE 9.3.5

The contact point B is the instantaneous point of rotation. The magnitudes of the velocities at A, C, and D are determined as

$$v_A = |AB|\omega = 2R\omega, \quad v_C = |CB|\omega = R\omega, \quad v_D = |DB|\omega.$$

The direction of one such velocity is perpendicular to the corresponding relative position vector.

If there is slipping between the disk and the surface, $v_B \neq 0$. In this case, $v_C \neq R\omega$ and the instantaneous point falls below B on the vertical line passing through points A and B.

9.3.2 Mass Moments of Inertia

The mass center G of a rigid body contained in a continuous domain Ω is defined by

$$\boldsymbol{r}_G = \frac{1}{m}\int_{\Omega} \boldsymbol{r}\, dm \tag{9.81}$$

where $m = \int_{\Omega} dm$ is the mass of the body, and $\boldsymbol{r}$ is the position vector of the differential mass dm; see Fig. 9.3.6. In Cartesian coordinates, the position of the mass center is given by

$$x_G = \frac{1}{m}\int_{\Omega} x\, dm, \quad y_G = \frac{1}{m}\int_{\Omega} y\, dm \tag{9.82}$$

The mass moments of inertia of the body Ω about x and y axes (Fig. 9.3.7) are

$$I_x = \int_{\Omega} x^2\, dm, \quad I_y = \int_{\Omega} y^2\, dm \tag{9.83}$$

The mass moments of inertia of the body Ω about pole O (z axis) is

$$I_z = \int_{\Omega} r^2\, dm \tag{9.84}$$

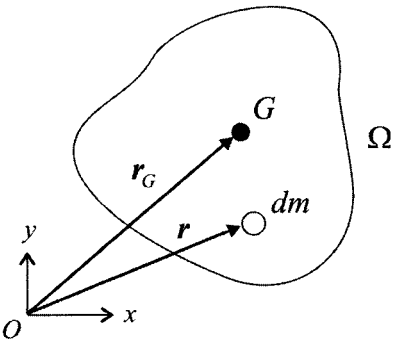


FIGURE 9.3.6 A rigid body of mass center G in a domain Ω.

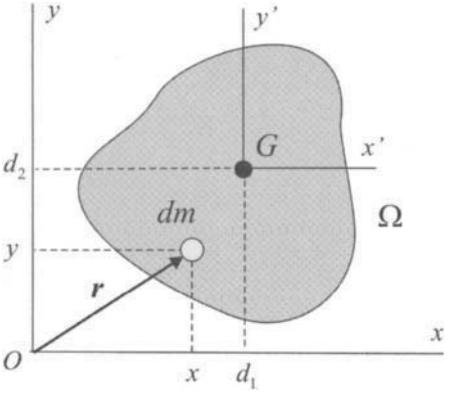


FIGURE 9.3.7

As can be seen,

$$I_z = I_x + I_y$$

The unit of I_x, I_y, and I_z is kg · m^2 in the SI system or slug-ft^2 in the U.S. system.

Now consider another coordinate system $Gx'y'$ whose origin is at the mass center G of the body and whose axes are parallel to those of Oxy; see Fig. 9.3.7. An axis that passes through the mass center is called a *central axis*. Let the moments of inertia of Ω in $Gx'y'$ be $\bar{I}_x$, $\bar{I}_y$, and $\bar{I}_z$. The moments of inertia in the two parallel coordinate systems are related by

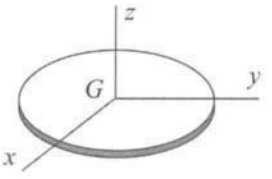
$$I_x = \bar{I}_x + md_1^2, \quad I_y = \bar{I}_y + md_2^2, \quad I_z = \bar{I}_z + md^2 \tag{9.85}$$

where $d^2 = d_1^2 + d_2^2$. Equation (9.85) is often referred to as *parallel-axis theorems*. Table 9.3.1 lists mass center and moments of inertia about the central axes for several homogeneous bodies.

The Toolbox of this chapter has a function `massmin` for computing the mass moments of inertia of the bodies listed in Table 9.3.1; see Window 3.1.

TABLE 9.3.1 Mass Moments of Inertia

1. Thin disk

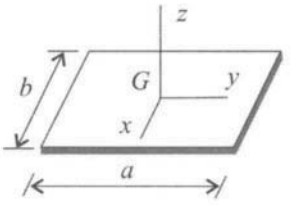


$$\bar{I}_x = \bar{I}_y = \frac{1}{4}mr^2$$

$$\bar{I}_z = \frac{1}{2}mr^2$$

`B_Spec = [1 m r]`

2. Rectangular plate



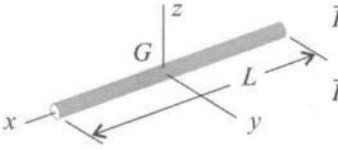
$$\bar{I}_x = \frac{1}{12}mb^2$$

$$\bar{I}_y = \frac{1}{12}ma^2$$

$$\bar{I}_z = \frac{1}{12}m(a^2 + b^2)$$

`B_Spec = [2 m a b]`

3. Slender rod

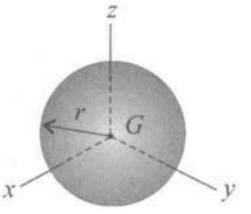


$$\bar{I}_x \approx 0$$

$$\bar{I}_y = \bar{I}_z = \frac{1}{12}mL^2$$

`B_Spec = [3 m L]`

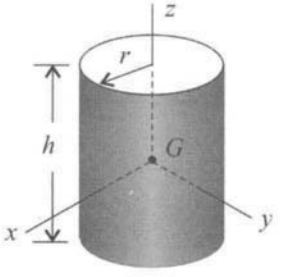
4. Sphere



$$\bar{I}_x = \bar{I}_y = \bar{I}_z = \frac{2}{5}mr^2$$

`B_Spec = [4 m r]`

5. Circular cylinder



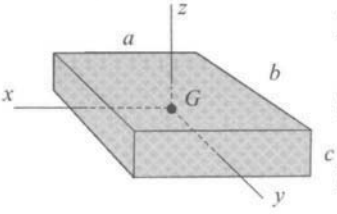
$$\bar{I}_x = \frac{1}{12}m(3r^2 + h^2)$$

$$\bar{I}_y = \bar{I}_x$$

$$\bar{I}_z = \frac{1}{2}mr^2$$

`B_Spec = [5 m r h]`

6. Rectangular prism



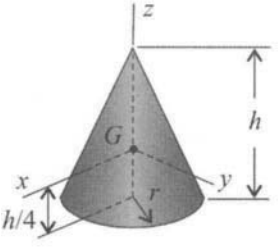
$$\bar{I}_x = \frac{1}{12}m(b^2 + c^2)$$

$$\bar{I}_y = \frac{1}{12}m(c^2 + a^2)$$

$$\bar{I}_z = \frac{1}{12}m(a^2 + b^2)$$

`B_Spec = [6 m a b c]`

7. Cone



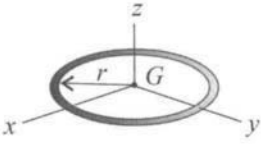
$$\bar{I}_x = \frac{3}{80}m(4r^2 + h^2)$$

$$\bar{I}_y = \bar{I}_x$$

$$\bar{I}_z = \frac{3}{10}mr^2$$

`B_Spec = [7 m r h]`

8. Hoop (ring)



$$\bar{I}_x = \bar{I}_y = \frac{1}{2}mr^2$$

$$\bar{I}_z = mr^2$$

`B_Spec = [8 m r]`

Window 3.1. Function massmin

MATLAB Function: **massmin**

Purpose:
To compute the mass moments of inertia of a given body about its central axis as shown in Table 9.3.1.

Synopsis:

```
massmin(B_Spec)
[Ix, Iy, Iz] = massmin(B_Spec)
```

Description:
`massmin(B_Spec)` computes the mass moments of inertia of a body, where `B_Spec` is a vector specifying the body by

```
B_Spec = [type_no  m  par1  par2 ...]
```

where `type_no` is an integer (from 1 to 8) identifying the shape type, `m` is the mass of the body, and `par1`, `par2`, ... are geometric parameters of the shape; see Table 9.3.1.

`[Ix, Iy, Iz] = massmin(B_Spec)` returns the computed mass moments of inertia $\bar{I}_x$, $\bar{I}_y$, and $\bar{I}_z$.

EXAMPLE 3.2

For the eight bodies in Table 9.3.1, assume that

$$m = 100, \quad r = 0.2, \quad a = 0.25, \quad b = 0.15, \quad c = 0.05, \quad L = 1, \quad h = 0.75.$$

The mass moments of inertia are computed by `massmin` and listed in the table below.

Cross-section	$\bar{I}_x$	$\bar{I}_y$	$\bar{I}_z$
1. Thin disk	1	1	2
2. Rectangular plate	0.1875	0.52083	0.70833
3. Slender rod	0	8.3333	8.3333
4. Sphere	1.6	1.6	1.6
5. Circular cylinder	5.6875	5.6875	2
6. Rectangular prism	0.20833	0.54167	0.70833
7. Cone	2.7094	2.7094	1.2
8. Hoop	2	2	4

9.3.3 Kinetics

Equations of Motion The plane motion of a rigid body Ω is described by the following equations about the mass center of the body

$$\begin{aligned} \sum \boldsymbol{F} &= m\boldsymbol{a}_G \\ \sum \boldsymbol{M}_G &= \bar{I}_z \boldsymbol{\alpha} \end{aligned} \tag{9.86}$$

where $\sum \boldsymbol{F}$ is the resultant force, $\sum \boldsymbol{M}_G$ is resultant moment about the mass center G, $m = \int_\Omega dm$ is the mass of the body, $\bar{I}_z$ is the mass moment of inertia about z axis passing through G, $\boldsymbol{a}_G$ is the acceleration of the mass center, and $\boldsymbol{\alpha} = \alpha \boldsymbol{k}$ is the angular acceleration vector. Equations (9.86) are written in scalar form in the Cartesian coordinates

$$\begin{aligned} \sum F_x &= m\bar{a}_x = m\ddot{x}_G \\ \sum F_y &= m\bar{a}_y = m\ddot{y}_G \\ \sum M_G &= \bar{I}_z \alpha = \bar{I}_z \ddot{\theta} \end{aligned} \tag{9.87}$$

where θ is the angular displacement of the body. Equations (9.86) and (9.87) show that the motion of a rigid body is completely defined by the resultant and moment resultant about G of the external forces acting on the body.

Sometimes, it is convenient to take moments of forces about a point, say A, that could be moving. Under the circumstances, the third equation of (9.87) can be replaced by

$$\sum M_A = I_G \ddot{\theta} + m\bar{a}_x y_A - m\bar{a}_y x_A \tag{9.88}$$

where $\sum M_A$ is the moment resultant about A of the external forces acting on the body, and x_A and y_A are the coordinates of A in the frame Gxy (see Fig. 9.3.8). Furthermore, if A is a fixed point or an instantaneous center of rotation (with zero velocity), Eq. (9.88) reduces to

$$\sum M_A = I_A \ddot{\theta} \tag{9.89}$$

where I_A is the mass moment of inertia about the axis through A, which, by the parallel axis theorems given in Eq. (9.85), is $I_A = \bar{I}_z + (x_A^2 + y_A^2)\, m$.

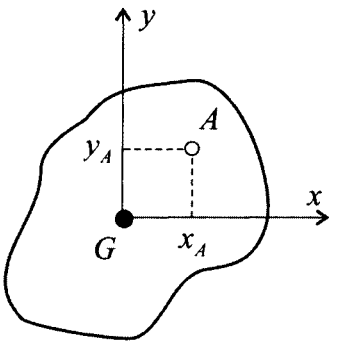


FIGURE 9.3.8

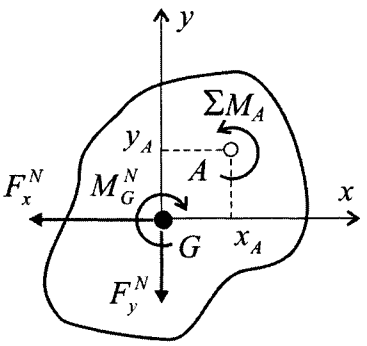


FIGURE 9.3.9 A rigid body under inertial forces.

D'Alembert's Principle for Rigid Bodies In this principle, Eqs. (9.86) are written as

$$\begin{aligned} \sum \boldsymbol{F} + \boldsymbol{F}^N &= 0 \\ \sum \boldsymbol{M}_G + \boldsymbol{M}_G^N &= 0 \end{aligned} \tag{9.90}$$

where $\boldsymbol{F}^N = -m\boldsymbol{a}_G$ is the *inertial force*, and $\boldsymbol{M}_G^N = -\bar{I}_z\,\boldsymbol{\alpha}$ is the *inertial moment* about G. Thus, the equations of motion for a rigid body are viewed as a result of balance of external forces and inertia forces.

As an example, D'Alembert's principle is applied to prove Eq. (9.88). Consideration of the equilibrium of the moments about A of external forces and inertial forces in Fig. 9.3.9 leads to

$$\sum M_A - M_G^N - F_x^N y_A + F_y^N x_A = 0$$

Substituting

$$F_x^N = m\bar{a}_x, \quad F_y^N = m\bar{a}_y, \quad M_G^N = \bar{I}_z\ddot{\theta}$$

into the previous equation yields Eq. (9.88).

Kinetic Energy The kinetic energy of a rigid body Ω in plane motion is defined as

$$T(t) = \int_{\Omega} \frac{1}{2}\boldsymbol{v}\cdot\boldsymbol{v}\,dm \tag{9.91}$$

It can be shown that

$$T(t) = \frac{1}{2}mv_G^2 + \frac{1}{2}\bar{I}_z\omega^2 \tag{9.92}$$

where $v_G = |\boldsymbol{v}_G|$ is the magnitude of the velocity of the mass center G of the body and ω is the angular velocity of the body. The first term of Eq. (9.92) is the kinetic energy of a translation with the mass center G; the second is the kinetic energy of a rotation about the axis through G.

The expression in Eq. (9.92) has the following two special cases:

Case 1. The body is in a translation ($\omega = 0$)

$$T(t) = \frac{1}{2} m v_G^2 \tag{9.93}$$

Case 2. The body is in a rotation about O, which is either a fixed point or an instantaneous center of rotation,

$$T(t) = \frac{1}{2} \bar{I}_O \omega^2 \tag{9.94}$$

where $\bar{I}_O = \bar{I}_z + md^2$, with $d = |OG|$ being the distance between O and G. Here $v_G = \omega d$ has been used to deduce Eq. (9.94) from Eq. (9.92).

Principle of Work and Energy For a rigid body in plane motion, the work W_{12} done by the external forces acting on the body equals the change in the kinetic energy; that is

$$W_{12} = \Delta T \tag{9.95}$$

If all the external forces are conservative,

$$W_{12} = -\Delta V \tag{9.96}$$

where V is the potential of the rigid body. Under the circumstances, the mechanical energy of the body conserved: $\Delta(T + V) = 0$, or

$$T_1 + V_1 = T_2 + V_2 \tag{9.97}$$

Linear and Angular Momentum The linear momentum of a rigid body Ω is

$$\boldsymbol{L}(t) = \int_{\Omega} \boldsymbol{v}\, dm = m\boldsymbol{v}_G \tag{9.98}$$

where $\boldsymbol{v}_G$ is the velocity of the mass center G, and Eq. (9.81) has been used. Differentiation of the momentum and use of Eq. (9.86), yields

$$\sum \boldsymbol{F} = \dot{\boldsymbol{L}} \tag{9.99}$$

It follows that

$$\int_{t_1}^{t_2} \sum \boldsymbol{F}\, dt = \boldsymbol{L}(t_2) - \boldsymbol{L}(t_1) \tag{9.100}$$

which is the principle of impulse and momentum for the body.

The angular momentum of the rigid body about G

$$\boldsymbol{H}_G(t) = \int_G \boldsymbol{r}' \times \boldsymbol{v} dm = \bar{I}_z \boldsymbol{\omega} = \bar{I}_z \omega \boldsymbol{k} \tag{9.101}$$

where $\boldsymbol{r}'$ is the position vector of dm that runs from G, and $\boldsymbol{\omega}$ is the angular acceleration of the body. By Eq. (9.86),

$$\sum \boldsymbol{M}_G = \dot{\boldsymbol{H}}_G(t) \tag{9.102}$$

Integration of the above equation gives another principle of impulse and momentum

$$\int_{t_1}^{t_2} \sum \boldsymbol{M}_G \, dt = \boldsymbol{H}_G(t_2) - \boldsymbol{H}_G(t_1) \tag{9.103}$$

If the body rotates about a fixed axis through O,

$$\int_{t_1}^{t_2} \sum M_O \, dt = I_O \omega_2 - I_O \omega_1 \tag{9.104}$$

9.3.4 Simulation of Dynamic Response

Following Sec. 9.2.4, the dynamic response of a rigid body in plane motion is determined by numerical integration. For this, Eq. (9.87) is rewritten in the state form

$$\dot{\eta}(t) = \boldsymbol{\Phi}(t, \eta(t)); \quad \eta(0) = \eta_0 \tag{9.105}$$

where

$$\begin{aligned} \eta(t) &= \begin{pmatrix} x_G & y_G & \theta & \dot{x}_G & \dot{y}_G & \dot{\theta} \end{pmatrix}^T \\ \boldsymbol{\Phi}(t, \eta(t)) &= \begin{pmatrix} \dot{x}_G & \dot{y}_G & \dot{\theta} & \dfrac{1}{m}\phi_1 & \dfrac{1}{m}\phi_2 & \dfrac{1}{\bar{I}_z}\phi_3 \end{pmatrix}^T \end{aligned} \tag{9.106}$$

the forcing functions

$$\phi_1(t, \eta(t)) = \sum F_x, \quad \phi_2(t, \eta(t)) = \sum F_y, \quad \phi_3(t, \eta(t)) = \sum M_G \tag{9.107}$$

and the initial condition vector

$$\eta_0 = \begin{pmatrix} x_G(0) & y_G(0) & \theta(0) & \dot{x}_G(0) & \dot{y}_G(0) & \dot{\theta}(0) \end{pmatrix}^T \tag{9.108}$$

Equation (9.105) is readily solved by the Runge-Kutta method given in Eq. (9.36).

The Toolbox of this chapter provides a function `rigid2d` for numerical simulation of the dynamic response of a rigid body in plane motion; see Window 3.2. Also, the Toolbox has another function `animotion2d` for animation of rigid body motion; see Window 3.3.

Window 3.2. Function `rigid2d`

MATLAB Function: **`rigid2d`**

Purpose:
To plot the plane motion of a rigid body by solving the state equation (9.105), based on the Runge-Kutta algorithm given in Eq. (9.36).

Window 3.2. Function `rigid2d` (*Continued*)

Synopsis:

```
rigid2d(m, Iz, 'ForceFn', eta0)
rigid2d(m, Iz, 'ForceFn', eta0, tf, npts)
[t, eta, KE, F] = rigid2d(m, Iz, 'ForceFn', eta0, tf, npts)
```

Description:

`rigid2d(m, Iz, 'ForceFn', eta0)` plots the trajectory of the mass center of the rigid body in two dimensions, the time histories of x_G, y_G, θ, the kinetic energy of the rigid body, and the resultant external forces and moment. Here `m` is the mass of the body, `Iz` is the mass moment of inertia about G, `ForceFn` is the name of a user-provided M-file function that describes the external forces ϕ_j in Eq. (9.107), and `eta0` is the vector of initial displacements and velocities given in Eq. (9.108). The user-provided function has the format

```
f = ForceFn(t, z)
```

where `t` is the time, `z` is the state vector defined in Eq. (9.106), and `f` is the vector of forcing functions ϕ_1, ϕ_2, and ϕ_3. If the body has zero initial disturbances, simply call `rigid2d(m, Iz, 'ForceFn')`. On the other hand, if no external force acts on the body, use `rigid2d(m, Iz, 0, eta0)` or `rigid2d(m, Iz, [ ], eta0)`.

`rigid2d(m, Iz, 'ForceFn', eta0, tf, npts)` plots the dynamic response of the rigid body with specified parameters `tf` and `npts`, where `tf` is total simulation time and `npts` is the number of time points in the region $0 \leq t \leq$ `tf`. The step size h of the Runge-Kutta algorithm is controlled by $h =$ `tf`/(`npts` $- 1$). By default, `tf` = 5 and `npts` = 1,001. If there is no external force, '`ForceFn`' is replaced by 0 or [ ]. If the body has zero initial disturbances, `eta0` is replaced by 0 or [ ].

`[t, eta, KE, F] = rigid2d(m, Iz, 'ForceFn', eta0, tf, npts)` returns the times of simulation in a vector `t`, the time history of the state vector in a six-row matrix `eta`, the kinetic energy of the body in a vector `KE`, and the external forces (ϕ_1, ϕ_2 and ϕ_3) in a three-row matrix `F`. The first three rows of `eta` contain the data of the displacements x_G, y_G, θ, while the last three rows the velocities $\dot{x}_G$, $\dot{y}_G$, $\dot{\theta}$. With these output variables, the motion of the rigid body can be plotted by the MATLAB function `plot`. For instance, `plot(t, eta (1,:), t, eta (2,:))` plots the trajectory of the mass center G in the 2-D space, and `plot(t, F(3,:))` plots the time history of the moment resultant $\phi_3 = \sum M_G$.

Window 3.3. Function `animotion2d`

MATLAB Function: **`animotion2d`**

Purpose:

To play the animated dynamic response of a rigid body.

Window 3.3. Function animotion2d (*Continued*)

Synopsis:

```
animotion2d(t, eta)
animotion2d(t, eta, IS_control, n_frames, frame_bounds)
[F, t_frame, z_frame] = animotion2d(t, eta, IS_control, n_frames,
frame_bounds)
```

Description:

`animotion2d(t, eta)` plays the animated dynamic response of the rigid body, where `t` is a vector of times and `eta` is a matrix about state vector. The `t` and `eta` are obtained by function `rigid2d`, such as `[t, eta] = rigid2d(m, Iz, 'ForceFn', eta0)`.

`animotion2d(t, eta, IS_control, n_frames, frame_bounds)` plays the animated response of the rigid body, with the specified parameters `IS_control`, `n_frames`, and `frame_bounds`. Here, `IS_control` is a control parameter with the following options:

`IS_control = 0`	play frames continuously, without trajectory preview
`IS_control = 1`	play frames continuously, with a trajectory preview
`IS_control = 2`	play frames one by one, without trajectory preview
`IS_control = 3`	play frames one by one, with a trajectory preview

`n_frames` is the total number of frames in the time period spanned by vector `t`; `frame_bounds = [xmin xmax ymin ymax]` sets up the bounds of frames. By default, `IS_control = 0`, `n_frames = 101`, and `frame_bounds` is automatically set up according to the matrix `eta`. The `IS_control`, `n_frames`, and `frame_bounds` can be replaced by `[ ]` as a placeholder for their default setup.

`[F, t_frame, z_frame] = animotion2d(t, eta, IS_control, n_frames, frame_bounds)` returns a set of frames of the motion of the rigid body in a matrix `F`, which later on can be played by the MATLAB function `movie(F)`, the times for the frames in a vector `t_frame`, and the response (x_G, y_G, θ) of the rigid body at the frame times in a three-row matrix `z_frame`.

In animation of the plane motion of a rigid body by function `animotion2d`, the symbol o— is used to indicate the location and orientation of the body. As illustrated in Fig. 9.3.10, when the rigid body moves along its trajectory, the circular part of the symbol tells the location (x_G, y_G) of the mass center of the body, and the bar tells the rotation (θ) of the body about the

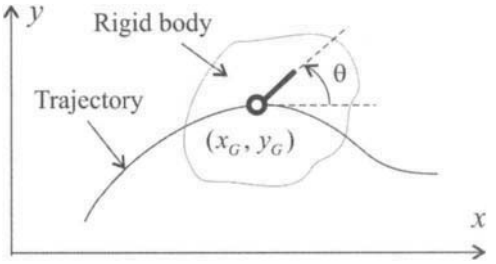


FIGURE 9.3.10 Illustration of the location and orientation of a body.

mass center. Here, the bar is any line embedded in the rigid body. Note that the aspect ratio of x to y needs to be 1:1, in order to show the real rotation angle.

The utility of `animotion2d` requires the data on the rigid body motion, which can be obtained by function `rigid2d`. Thus, before `animotion2d` can be used, `rigid2d` must be called first. See Example 3.3 for a demonstration.

EXAMPLE 3.3

In Fig. 9.3.11(a), a slender rod of mass $m = 1$ kg and length $l = 0.25$ m is in an upright position and at an elevation of 2.2 m, when it is thrown with initial velocity $\boldsymbol{v}_0 = 25\boldsymbol{i}+30\boldsymbol{j}$ m/s and initial angular velocity $\dot{\theta}_0 = -10$rad/s. The rod is constrained by a stretchable string of initial length $L_0 = 40$ m, which is connected to the mass center G of the rod and the reference point O. The string, when stretched, exerts a force of magnitude $F_s = k_s(L - L_0)$ on the rod, where $L = |OG|$ and $k_s = 2$ N/m. A free-body diagram of the rod is shown in Fig. 9.3.11(b), where, besides the gravity and string tension force, an aerodynamic drag $F_{\text{drag}} = 0.0005v_G^2$ and an air damping moment $M_{\text{damp}} = 0.001\omega|\omega|$ N·m also act on the rod. Here, v_G is the velocity of G and ω is the angular velocity of the body.

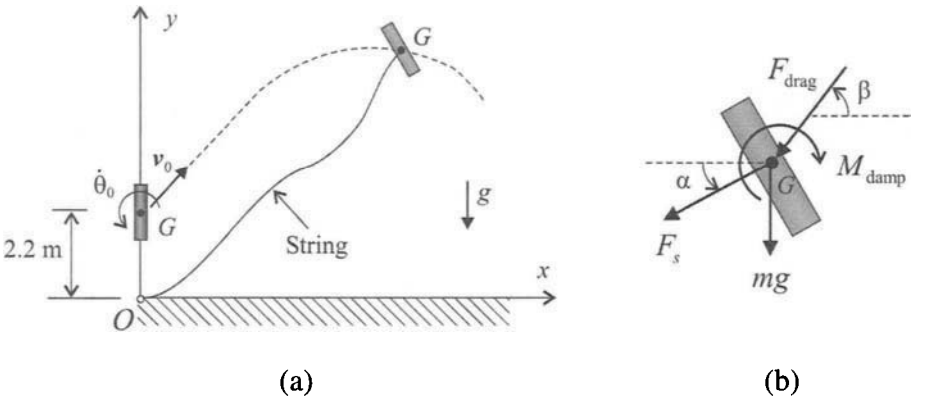


FIGURE 9.3.11

According to the above description, the differential equations of motion for the rod are

$$m\ddot{x}_G = \phi_1 = -F_s\cos\alpha - 0.0005\left(\dot{x}_G^2 + \dot{y}_G^2\right)\cos\beta$$

$$m\ddot{y}_G = \phi_2 = -F_s\sin\alpha - 0.0005\left(\dot{x}_G^2 + \dot{y}_G^2\right)\sin\beta - mg$$

$$\bar{I}_z\ddot{\theta} = \phi_3 = -0.001\dot{\theta}|\dot{\theta}|$$

where

$$F_s = \begin{cases} 0, & \text{for } \sqrt{x_G^2 + y_G^2} \le L_0 \\ k_s\left(\sqrt{x_G^2 + y_G^2} - L_0\right), & \text{for } \sqrt{x_G^2 + y_G^2} > L_0 \end{cases}$$

$$\cos\alpha = \frac{x_G}{\sqrt{x_G^2 + y_G^2}}, \quad \sin\alpha = \frac{y_G}{\sqrt{x_G^2 + y_G^2}}$$

$$\cos\beta = \frac{\dot{x}_G}{\sqrt{\dot{x}_G^2 + \dot{y}_G^2}}, \quad \sin\beta = \frac{\dot{y}_G}{\sqrt{\dot{x}_G^2 + \dot{y}_G^2}}$$

$$\tilde{I}_z = ml^2/12 = 0.0052 \text{ kg} \cdot \text{m}^2.$$

The initial conditions of the rod are

$$x_G(0) = 0, \quad \dot{x}_G(0) = 25 \text{ m/s}; \quad y_G(0) = 2.2 \text{ m},$$

$$\dot{y}_G(0) = 30\ m/s; \quad \theta(0) = \frac{\pi}{2} \text{ rad}, \quad \dot{\theta}(0) = -10 \text{ rad/s}$$

To solve the above equations of the motion by function `rigid2d`, an M-file function `force_exm33` for describing the forcing functions ϕ_j is created:

```
% function named force_exm33
function f = force_exm33(tt, a)
m = 1; L0 = 40;
x = a(1); y = a(2);
L = sqrt(x^2 + y^2);
if L0 >= L
    fs = [0; 0; 0];
else
    Fs1 = 2*(L - L0);
    fs = [-Fs1*x/L; -Fs1*y/L; 0];
end
vx = a(4); vy = a(5); w = a(6);
v = sqrt(vx^2 + vy^2);
f = [-0.0005*v*vx; -0.0005*v*vy-m*9.81; -0.001*w*abs(w)] + fs;
```

(See Appendix B.4 for how to generate an M-file.) By the MATLAB commands

```
>> m = 1; Iz = 1/12*m*(0.25)^2
>> eta0 = [0 2.2 pi/2 25 30 -10]';
>> [t, eta] = rigid2d(m, Iz, 'force_exm33', eta0, 3.5);
```

the trajectory, kinetic energy, and time response of the rigid body, and the time histories of the external forces are plotted in Figs. 9.3.12, for the first 3.5 seconds.

From Figs. 9.3.12(a) and (d), the string tension force does not appear until the distance between G and O is beyond 40 m, which is the length of an undeformed string. As the rod flies further away, the string tension force increases. Around $t = 2.1$ s, the tension force is strong enough to prevent the rod from flying further away. It is at this time the kinetic energy of the rod reaches its lowest point; see Fig. 9.3.12(b). As the rod is being pulled back by the string tension, the distance between G and O decreases. About $t = 3.2$ s, the string tension force disappears as $|OG|$ is less then 40 m. Afterwards, the rod experiences a free fall under the influence of air drag and damping, and eventually lands on the ground approximately at $x = 32$ m. During its motion, the rod keeps

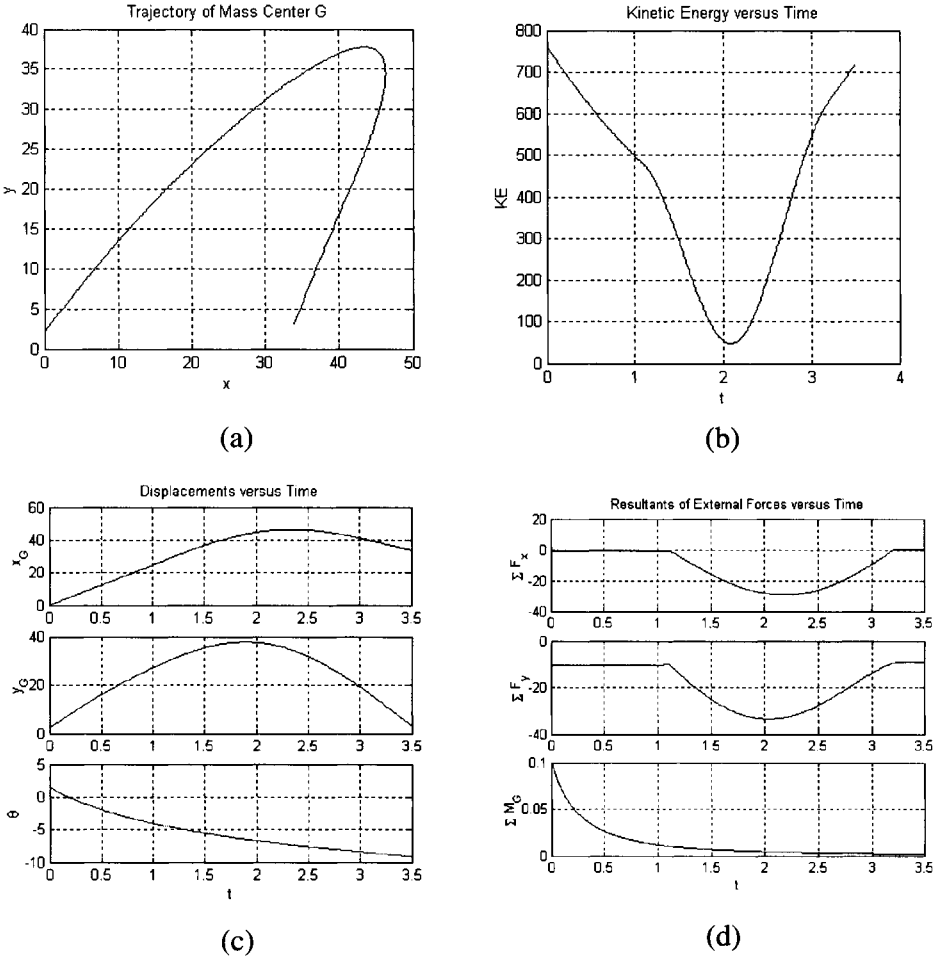


FIGURE 9.3.12

spinning clockwise about its mass center, which is caused by the initial angular velocity. However, the air damping moment gradually slows the rotation down, as indicated by the moment curve in Fig. 9.3.12(d).

To see the animated motion of the rod, simply type

```
>> animotion2d(t, eta)
```

where `t` and `eta` are the output arguments given by function `rigid2d`. Shown in Figure 9.3.13 are six frames of the animated motion that are obtained by

```
>> animotion2d(t, eta, 3, 6)
```

The symbol o— in the figure indicates the location of the mass center and rotation of the rod, as described in Fig. 9.3.10.

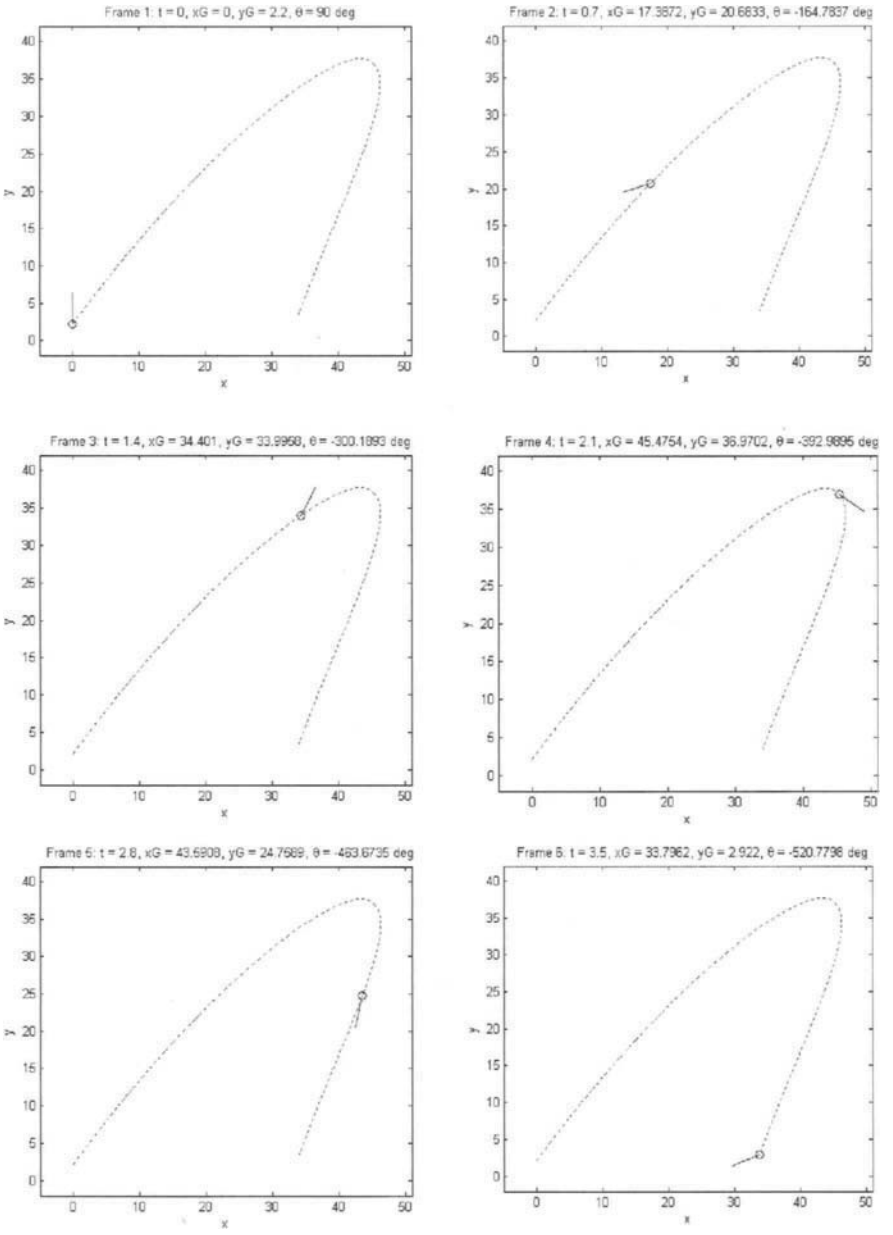


FIGURE 9.3.13

9.4 Rigid Body Dynamics in Three Dimensions

9.4.1 Kinematics

Rotation about a Fixed Axis In Fig. 9 4.1, a rigid body rotates about axis *A-A* through the reference point O, with angular velocity $\boldsymbol{\omega}$. The velocity of a point of the body is

$$\boldsymbol{v} = \frac{d\boldsymbol{r}}{dt} = \boldsymbol{\omega} \times \boldsymbol{r} \tag{9.109}$$

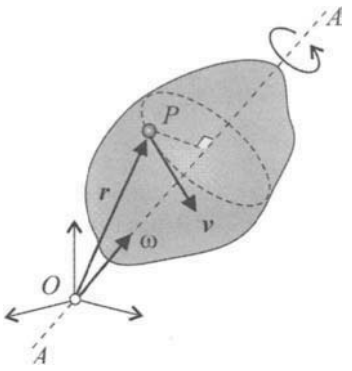


FIGURE 9.4.1 A rigid body rotating about a fixed axis.

where $\boldsymbol{r}$ is the position vector of the point. The acceleration of that point is given by

$$\boldsymbol{a} = \frac{d\boldsymbol{v}}{dt} = \boldsymbol{\alpha} \times \boldsymbol{r} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r}) \tag{9.110}$$

where $\boldsymbol{\alpha} = \dot{\boldsymbol{\omega}}$ is the angular acceleration of the body. The angular velocity $\boldsymbol{\omega}$ and angular acceleration $\boldsymbol{\alpha}$ are vectors that follow the parallelogram rule in vector addition. However, *finite* rotations of a rigid body do not obey the rule of vector addition.

Relative Motion in a Translating Reference Frame In Fig. 9.4.2, A and B are any two points of a rigid body, $OXYZ$ is a fixed reference frame, and $AX'Y'Z'$ is a translating reference frame attached to A. The motion of $AX'Y'Z'$ is the same as that of A and the X'-, Y'-, and Z'-axes have fixed orientation. The motion of A is related to that of B by

$$\begin{aligned} \boldsymbol{v}_B &= \boldsymbol{v}_A + \boldsymbol{v}_{B/A} \\ \boldsymbol{a}_B &= \boldsymbol{a}_A + \boldsymbol{a}_{B/A} \end{aligned} \tag{9.111}$$

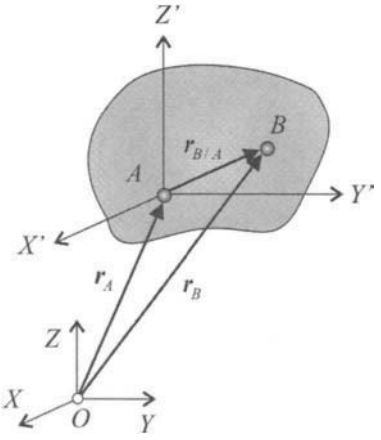


FIGURE 9.4.2 A rigid body in fixed and translating frames.

where $\boldsymbol{r}_{B/A}$ is the position vector of B relative to A, $\boldsymbol{v}_{B/A}$ and $\boldsymbol{a}_{B/A}$ are the velocity and acceleration that describe a rotation of B relative to A. According to Eqs. (9.109) and (9.110), the relative velocity and acceleration can be expressed as

$$\begin{aligned} \boldsymbol{v}_{B/A} &= \dot{\boldsymbol{r}}_{B/A} = \boldsymbol{\omega} \times \boldsymbol{r}_{B/A} \\ \boldsymbol{a}_{B/A} &= \dot{\boldsymbol{v}}_{B/A} = \boldsymbol{\alpha} \times \boldsymbol{r}_{B/A} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r}_{B/A}) \end{aligned} \tag{9.112}$$

where $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ are the angular velocity and acceleration of the body, respectively.

Time Derivatives of Vectors in a Moving Reference Consider two reference frames in Fig. 9.4.3: a fixed frame XYZ, and a moving frame xyz that moves arbitrarily relative to the fixed frame with translational velocity $\dot{\boldsymbol{R}}$ and angular velocity $\boldsymbol{\Omega}$. Let P be a particle moving in the space. The position vector of P in XYZ is $\boldsymbol{r}$ and that in xyz is $\boldsymbol{\rho}$. Note that

$$\boldsymbol{r} = \boldsymbol{R} + \boldsymbol{\rho} \tag{9.113}$$

Let $\left(\frac{d\boldsymbol{A}}{dt}\right)_{XYZ}$ be the time derivative of a vector $\boldsymbol{A}$ from the fixed frame, which is a time rate of change seen by an observer in the fixed frame. Let $\left(\frac{d\boldsymbol{A}}{dt}\right)_{xyz}$ be the time derivative of $\boldsymbol{A}$ from the moving frame, which is a time rate of change seen by an observer on the moving frame. It can be shown that

$$\left(\frac{d\boldsymbol{A}}{dt}\right)_{XYZ} = \left(\frac{d\boldsymbol{A}}{dt}\right)_{xyz} + \boldsymbol{\Omega} \times \boldsymbol{A} \tag{9.114}$$

Apply Eq. (9.114) to Eq. (9.113) to obtain

$$\boldsymbol{v}_{XYZ} = \boldsymbol{v}_{xyz} + \dot{\boldsymbol{R}} + \boldsymbol{\Omega} \times \boldsymbol{\rho} \tag{9.115}$$

$$\boldsymbol{a}_{XYZ} = \boldsymbol{a}_{xyz} + \ddot{\boldsymbol{R}} + 2\boldsymbol{\Omega} \times \boldsymbol{v}_{xyz} + \dot{\boldsymbol{\Omega}} \times \boldsymbol{\rho} + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times \boldsymbol{\rho}) \tag{9.116}$$

where $\dot{\boldsymbol{R}} = \left(\frac{d\boldsymbol{R}}{dt}\right)_{XYZ}$, $\dot{\boldsymbol{\Omega}} = \left(\frac{d\boldsymbol{\Omega}}{dt}\right)_{XYZ}$

$$\boldsymbol{v}_{XYZ} = \left(\frac{d\boldsymbol{r}}{dt}\right)_{XYZ} = \text{velocity of } P \text{ relative to the fixed frame } XYZ$$

$$\boldsymbol{v}_{xyz} = \left(\frac{d\boldsymbol{\rho}}{dt}\right)_{xyz} = \text{velocity of } P \text{ relative to the moving frame } xyz$$

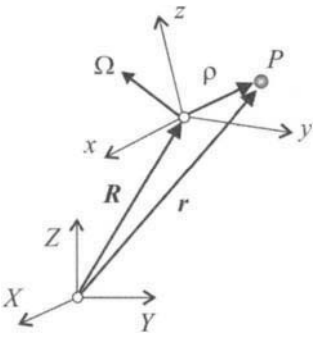


FIGURE 9.4.3 Two reference frames.

$$\boldsymbol{a}_{XYZ} = \left(\frac{d^2\boldsymbol{r}}{dt^2}\right)_{XYZ} = \text{acceleration of } P \text{ relative to the fixed frame XYZ}$$

$$\boldsymbol{a}_{xyz} = \left(\frac{d^2\boldsymbol{\rho}}{dt^2}\right)_{xyz} = \text{acceleration of } P \text{ relative to the moving frame } xyz$$

The term $2\boldsymbol{\Omega} \times \boldsymbol{v}_{xyz}$ in Eq. (9.116) is called *Coriolis acceleration.*

EXAMPLE 4.1

A tube containing a ball is attached to a vertical shaft with a clevis; see Fig. 9.4.4, where $Oxyz$ is a fixed frame and $Ox'y'z'$ is a rotating frame attached to the tube with the x' axis being horizontal all the time. The shaft rotates at a constant speed ω_0. The tube makes a tilting angle $\beta(t)$ from the z axis. The ball is moving inside the tube with a distance $b(t) = |\text{OB}|$ from point O. At the moment, the axes x and x' coincide and the tube is in the plane Oyz. Determine the velocity and acceleration of the ball.

Solution. Let $\boldsymbol{I}$, $\boldsymbol{J}$, and $\boldsymbol{K}$ be the unit vectors of the fixed frame. Let $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ be the unit vectors of the rotating frame. The angular velocity of the moving frame is

$$\boldsymbol{\Omega} = \dot{\beta}\boldsymbol{i} + \omega_0\boldsymbol{K}$$

By Eq. (9.114), the angular acceleration of the moving frame is

$$\dot{\boldsymbol{\Omega}} = \ddot{\beta}\boldsymbol{i} + \boldsymbol{\Omega} \times \dot{\beta}\boldsymbol{i} = \ddot{\beta}\boldsymbol{i} - \dot{\beta}\boldsymbol{i} \times \omega_0\boldsymbol{K}$$

The position vector of the ball and its derivatives with respect to the rotating frames are

$$\boldsymbol{\rho} = b\boldsymbol{j}, \quad \left(\frac{d\boldsymbol{\rho}}{dt}\right)_{x'y'z'} = \dot{b}\boldsymbol{j}, \quad \left(\frac{d^2\boldsymbol{\rho}}{dt^2}\right)_{x'y'z'} = \ddot{b}\boldsymbol{j}$$

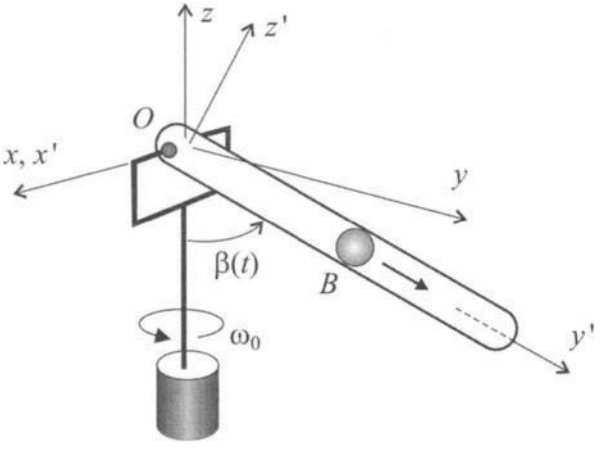


FIGURE 9.4.4

The velocity of the ball relative to the fixed frame is obtained as

$$\begin{aligned}\boldsymbol{v}_{xyz} &= \left(\frac{d\boldsymbol{\rho}}{dt}\right)_{x'y'z'} + \boldsymbol{\Omega} \times \boldsymbol{\rho} = \dot{b}\boldsymbol{j} + (\dot{\beta}\boldsymbol{i} + \omega_0 \boldsymbol{K}) \times b\boldsymbol{j} \\ &= -b\omega_0 \sin\beta\, \boldsymbol{i} + \dot{b}\boldsymbol{j} + b\dot{\beta}\boldsymbol{k}\end{aligned}$$

where $\boldsymbol{K} \times \boldsymbol{j} = -\sin\beta\, \boldsymbol{i}$ has been used. At the moment of interest

$$\boldsymbol{i} = \boldsymbol{I}, \quad \boldsymbol{j} = \sin\beta\, \boldsymbol{J} - \cos\beta\, \boldsymbol{K}, \quad \boldsymbol{k} = \cos\beta\, \boldsymbol{J} + \sin\beta\, \boldsymbol{K}$$

Thus, the velocity of the ball can also be expressed by the components in the fixed frame

$$\boldsymbol{v}_{xyz} = -b\omega_0 \sin\beta\, \boldsymbol{I} + (\dot{b}\sin\beta + b\dot{\beta}\cos\beta)\boldsymbol{J} + (b\dot{\beta}\sin\beta - \dot{b}\cos\beta)\boldsymbol{K}$$

The acceleration of the ball is derived as follows

$$\begin{aligned}\boldsymbol{a}_{xyz} &= \left(\frac{d^2\boldsymbol{\rho}}{dt^2}\right)_{x'y'z'} + 2\boldsymbol{\Omega} \times \left(\frac{d\boldsymbol{\rho}}{dt}\right)_{x'y'z'} + \dot{\boldsymbol{\Omega}} \times \boldsymbol{\rho} + \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times \boldsymbol{\rho}) \\ &= \ddot{b}\boldsymbol{j} + 2(\dot{\beta}\boldsymbol{i} + \omega_0\boldsymbol{K}) \times \dot{b}\boldsymbol{j} + (\ddot{\beta}\boldsymbol{i} - \dot{\beta}\boldsymbol{i} \times \omega_0\boldsymbol{K}) \times b\boldsymbol{j} + (\dot{\beta}\boldsymbol{i} + \omega_0\boldsymbol{K}) \\ &\quad \times b(-\omega_0 \sin\beta\, \boldsymbol{i} + \dot{\beta}\boldsymbol{k}) \\ &= -2\omega_0(\dot{b}\sin\beta + b\dot{\beta}\cos\beta)\boldsymbol{i} + \left(\ddot{b} - b\dot{\beta}^2 - b\omega_0^2 \sin^2\beta\right)\boldsymbol{j} \\ &\quad + \left(b\ddot{\beta} + 2\dot{b}\dot{\beta} - b\omega_0^2 \sin\beta\cos\beta\right)\boldsymbol{k}\end{aligned}$$

where $\boldsymbol{K} \times \boldsymbol{j} = -\sin\beta\boldsymbol{i}$ and $\boldsymbol{J} \times \boldsymbol{j} = \boldsymbol{K} \times \boldsymbol{k} = -\cos\beta\boldsymbol{i}$ have been used. The acceleration can also be expressed by the components in the fixed frame as follows:

$$\begin{aligned}\boldsymbol{a}_{xyz} &= -2\omega_0(\dot{b}\sin\beta + b\dot{\beta}\cos\beta)\boldsymbol{I} + \left((\ddot{b} - b\dot{\beta}^2 - b\omega_0^2)\sin\beta + (b\ddot{\beta} + 2\dot{b}\dot{\beta})\cos\beta\right)\boldsymbol{J} \\ &\quad + \left((b\ddot{\beta} + 2\dot{b}\dot{\beta})\sin\beta + (b\dot{\beta}^2 - \ddot{b})\cos\beta\right)\boldsymbol{K}\end{aligned}$$

9.4.2 Inertia Properties

Mass Center The mass center G of a rigid body $\mathscr{B}$ is

$$\boldsymbol{r}_G = \frac{1}{m}\int_{\mathscr{B}} \boldsymbol{r}\, dm \tag{9.117}$$

where $m = \int_{\mathscr{B}} dm$ is the mass of the body, and $\boldsymbol{r}$ is the position vector of the differential mass dm related to a fixed frame.

Linear Momentum The linear momentum of a rigid body $\mathscr{B}$ is

$$\boldsymbol{L}(t) = \int_{\mathscr{B}} \boldsymbol{v}\, dm = m\boldsymbol{v}_G \tag{9.118}$$

where $\boldsymbol{v}_G = \dot{\boldsymbol{r}}_G$ is the velocity of the mass center G.

Angular Momentum The angular momentum of a rigid body $\mathscr{B}$ about its mass center G is similar to that given in Eq. (9.47), namely,

$$\boldsymbol{H}_G(t) = \int_{\mathscr{B}} \boldsymbol{r}' \times \boldsymbol{v}\, dm \tag{9.119}$$

where $\boldsymbol{v}$ is the absolution velocity of dm relative to a fixed frame $OXYZ$, and $\boldsymbol{r}'$ is the position vector of dm relative to a translating frame $Gxyz$ that moves with G; see Fig. 9.4.5. According to Eqs. (9.111) and (9.112),

$$\boldsymbol{v} = \boldsymbol{v}_G + \boldsymbol{v}' = \boldsymbol{v}_G + \boldsymbol{r}' \times \boldsymbol{\omega} \tag{9.120}$$

where $\boldsymbol{v}_G$ is the velocity of G, $\boldsymbol{v}'$ is the relative velocity of dm with respect to $Gxyz$, and $\boldsymbol{\omega}$ is the angular velocity of the body. Substitute Eq. (9.120) into Eq. (9.119) to obtain

$$\boldsymbol{H}_G = \int_{\mathscr{B}} \boldsymbol{r}' \times (\boldsymbol{r}' \times \boldsymbol{\omega})\, dm \tag{9.121}$$

where $\int_{\mathscr{B}} \boldsymbol{r}'\, dm = 0$ has been used. By writing

$$\boldsymbol{r}' = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}, \quad \boldsymbol{\omega} = \omega_x \boldsymbol{i} + \omega_y \boldsymbol{j} + \omega_z \boldsymbol{k} \tag{9.122}$$

Eq. (9.121) is reduced to

$$\boldsymbol{H}_G = H_x \boldsymbol{i} + H_y \boldsymbol{j} + H_z \boldsymbol{k} \tag{9.123}$$

where

$$\begin{pmatrix} H_x \\ H_y \\ H_z \end{pmatrix} = \begin{bmatrix} \bar{I}_{xx} & -\bar{I}_{xy} & -\bar{I}_{xz} \\ -\bar{I}_{yx} & \bar{I}_{yy} & -\bar{I}_{yz} \\ -\bar{I}_{zx} & -\bar{I}_{zy} & \bar{I}_{zz} \end{bmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{9.124}$$

with

$$\bar{I}_{xx} = \int_{\mathscr{B}} (y^2 + z^2)dm, \quad \bar{I}_{yy} = \int_{\mathscr{B}} (z^2 + z^2)dm, \quad \bar{I}_{zz} = \int_{\mathscr{B}} (x^2 + y^2)dm \tag{9.125a}$$

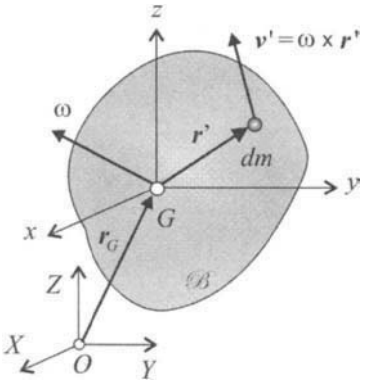


FIGURE 9.4.5 A rigid body with a translating frame $Gxyz$ that moves with the mass center G.

which are the moments of inertia of the body, and

$$\bar{I}_{xy} = \bar{I}_{yx} = \int_{\mathscr{B}} xydm, \quad \bar{I}_{yz} = \bar{I}_{zy} = \int_{\mathscr{B}} yzdm, \quad \bar{I}_{zx} = \bar{I}_{xz} = \int_{\mathscr{B}} zxdm \tag{9.125b}$$

which are the products of inertia. The unit of moments and products of inertia is kg·m^2 in the SI system or slug-ft^2 in the U.S. system. The symmetric matrix in Eq. (9.124) defines an inertia tensor of the rigid body.

The angular moment of a rigid body about any given point O is defined as

$$\boldsymbol{H}_O = \int_{\mathscr{B}} \boldsymbol{r} \times \boldsymbol{v} dm \tag{9.126}$$

where $\boldsymbol{r}$ is the position vector of dm that runs from O. Substitution of $\boldsymbol{r} = \boldsymbol{r}_G + \boldsymbol{r}'$ (see Fig. 9.4.5) into Eq. (9.126) and use of Eq. (9.118) yields

$$\boldsymbol{H}_O = \boldsymbol{H}_G + \boldsymbol{r}_G \times m\boldsymbol{v}_G \tag{9.127}$$

This indicates that $\boldsymbol{H}_O$ is the sum of $\boldsymbol{H}_G$ and the moment of the linear momentum $m\boldsymbol{v}_G$, as shown in Fig. 9.4.6.

If the rigid body rotates about a fixed point O with angular velocity $\boldsymbol{\omega}$, the velocity $\boldsymbol{v}$ of dm becomes $\boldsymbol{v} = \boldsymbol{r} \times \boldsymbol{\omega}$. In this case,

$$\boldsymbol{H}_O(t) = \int_{\mathscr{B}} \boldsymbol{r} \times (\boldsymbol{r} \times \boldsymbol{\omega})\, dm \tag{9.128}$$

Note that Eqs. (9.121) and (9.128) have the same format. It follows that

$$\boldsymbol{H}_O = H_{O,x}\boldsymbol{i} + H_{O,y}\boldsymbol{j} + H_{O,z}\boldsymbol{k} \tag{9.129}$$

with

$$\begin{pmatrix} H_{O,x} \\ H_{O,y} \\ H_{O,z} \end{pmatrix} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{9.130}$$

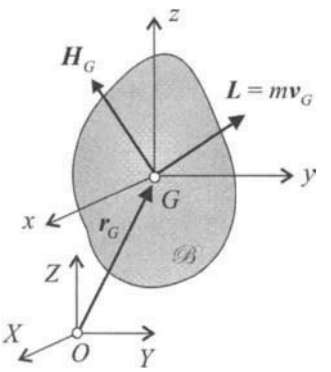


FIGURE 9.4.6 The angular moment of a body about a point.

where the moments of inertia and products of inertia are computed with respect to frame *OXYZ* that is centered at *O*.

Transformation of Inertia Properties Consider Fig. 9.4.5 again, where frames *OXYZ* and *Gxyz* have parallel axes. Let the coordinates of *G* in frame *OXYZ* be (x_G, y_G, z_G). The moments and products of inertia at point *O* can be expressed by those at *G* as follows

$$I_{xx} = \bar{I}_{xx} + m\left(y_G^2 + z_G^2\right), \quad I_{yy} = \bar{I}_{yy} + m\left(z_G^2 + x_G^2\right), \quad I_{zz} = \bar{I}_{zz} + m\left(x_G^2 + y_G^2\right)$$
$$I_{xy} = \bar{I}_{xy} + mx_G y_G, \quad I_{yz} = \bar{I}_{yz} + my_G z_G, \quad I_{zx} = \bar{I}_{zx} + mz_G x_G \tag{9.131}$$

which is an extension of the parallel-axis theorems given in Eq. (9.85).

Now consider two sets of orthogonal axes (x, y, z) and (x', y', z') at point *P*; see Fig. 9.4.7. *P* could be the mass center *G* or any other point. Denote the angles between (x, y, z) and (x', y', z') by the following:

$\alpha_{x'x}$ = the angle between the axes x' and x

$\alpha_{x'y}$ = the angle between the axes x' and y

$\alpha_{z'x}$ = the angle between the axes z' and x

.....

Define the matrix of inertia associated with frame *Pxyz* by

$$[I_M] = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{9.132}$$

Define the matrix of inertia associated with frame *Px'y'z'* by

$$[I'_M] = \begin{bmatrix} I_{x'x'} & -I_{x'y'} & -I_{x'z'} \\ -I_{y'x'} & I_{y'y'} & -I_{y'z'} \\ -I_{z'x'} & -I_{z'y'} & I_{z'z'} \end{bmatrix} \tag{9.133}$$

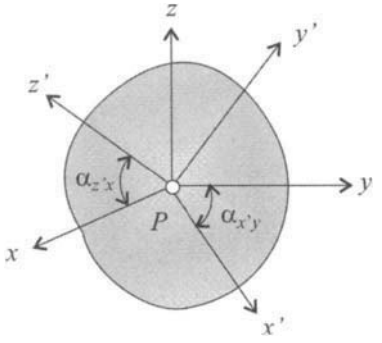


FIGURE 9.4.7 Two sets of orthogonal axes.

where $I_{x'x'} = \int_{\mathscr{B}} (y'^2 + z'^2) dm$, $I_{x'y'} = \int_{\mathscr{B}} x'y'\, dm$, etc. Both $[I_M]$ and $[I'_M]$ are symmetric matrices. The two matrices of inertia are related by the following transformation formula

$$[I'_M] = [D]^T [I_M] [D] \tag{9.134}$$

where $[D]$ is the matrix of direction cosines

$$[D] = \begin{bmatrix} \cos\alpha_{x'x} & \cos\alpha_{y'x} & \cos\alpha_{z'x} \\ \cos\alpha_{x'y} & \cos\alpha_{y'y} & \cos\alpha_{z'y} \\ \cos\alpha_{x'z} & \cos\alpha_{y'z} & \cos\alpha_{z'z} \end{bmatrix} \tag{9.135}$$

The matrix $[D]$ represents a coordinate transformation due to rotation. It can be shown that

$$[D]^T [D] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.136}$$

Principal Moments of Inertia Proper selection of a matrix $[D]$ of direction cosines can yield zero products of inertia; that is, $I_{x'y'} = 0$, $I_{y'z'} = 0$, $I_{z'x'} = 0$. In this case, $I_{x'x'}$, $I_{y'y'}$, $I_{z'z'}$ are called the *principal moments of inertia* of the rigid body; x', y', and z' are called the *principal axes of inertia*. The matrix of inertia associated with $x'y'z'$ is of diagonal form

$$[I'_M] = \begin{bmatrix} I_{x'x'} & 0 & 0 \\ 0 & I_{y'y'} & 0 \\ 0 & 0 & I_{z'z'} \end{bmatrix} \tag{9.137}$$

Table 9.3.1 lists the principal moments of inertia about mass center for eight homogeneous bodies.

Given a matrix $[I_M]$ of inertia associated with frame $Pxyz$, the principal axes and principal moments of inertia are the solutions of the following eigenvalue problem

$$[I_M] \{u\} = \lambda \{u\} \tag{9.138}$$

where λ is the eigenvalue and $\{u\}$ is the associate eigenvector. This problem has three eigenvalues $\lambda_1, \lambda_2, \lambda_3$, which are the principal moments of inertia. The associate eigenvectors $\{u_1\}, \{u_2\}, \{u_3\}$, which are normalized by

$$\{u_j\}^T \{u_j\} = 1, \quad j = 1, 2, 3 \tag{9.139}$$

form the direction cosine matrix for the principal axes by

$$[D] = \begin{bmatrix} \{u_1\} & \{u_2\} & \{u_3\} \end{bmatrix}. \tag{9.140}$$

The condition (9.139) and the orthogonality of $\{u_j\}$ imply Eq. (9.136). Thus, the components of the eigenvectors are the direction cosines given in Eq. (9.135).

The Toolbox of this chapter contains a function `inertiaTF` for computing moments and products of inertia under coordinate transformation due to rotation, and a function `prinertia` for determining principal moments of inertia and principal axes of inertia; see Windows 4.1 and 4.2.

Window 4.1. Function `inertiaTF`

MATLAB Function: **`inertiaTF`**

Purpose:
To compute the moments and products of inertia in a new frame by Eq. (9.134).

Synopsis:

```
inertiaTF(IM_old, D)
IM_new = inertiaTF(IM_old, D)
```

Description:
`inertiaTF(IM_old, D)` computes the moments and products of inertia associated with the axes x', y', and z', where `IM_old` is the given matrix $[I_M]$ of inertia associated with the axes x, y, and z as defined by Eq. (9.132), and `D` is the matrix of direction cosines as defined by Eq. (9.135). Function `inertiaTF` also shows the axes x', y', and z' in a three-dimensional plot.

`IM_new = inertiaTF(IM_old, D)` returns the computed matrix of inertia $[I'_M]$ in a matrix `IM_new`, which is associated with the axes x', y', and z' as defined by Eq. (9.133).

Window 4.2. Function `prinertia`

MATLAB Function: **`prinertia`**

Purpose:
To determine principal moments of inertia and principal axes of inertia.

Synopsis:

```
prinertia(IM_matrix)
[I_prin, A_Prin] = prinertia(IM_matrix)
```

Description:
`prinertia(IM_matrix)` computes the principal moments of inertia and principal axes, for a given matrix of inertia that are stored in `IM_matrix` according to Eq. (9.132). Function `prinertia` also shows the principal axes in a three-dimensional plot.

`[I_prin, A_Prin] = prinertia(IM_matrix)` returns the principal moments of inertia in a vector `I_prin`, and the direction cosines that define the principal axes in a matrix `A_Prin`.

EXAMPLE 4.2

The rigid body in Fig. 9.4.8 is composed of three uniform slender rods welded together. The mass per unit length of each rod is $\rho = 2$ kg/m. Compute the moments and products of inertia of the body associated with frame $Oxyz$ and determine the principal moments

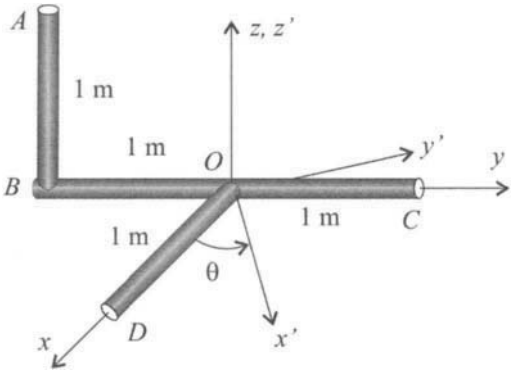


FIGURE 9.4.8

of inertia and the principal axes of inertia. Also, compute the moments and products of inertia of the body in the frame $Ox'y'z'$, which is obtained by rotating the z-axis with an angle $\theta = 25°$ with x'- and y'-axes in the Oxy plane.

Solution. By Table 9.3.1 and Eq. (9.131), the moments and products of inertia of each rod about O are computed and listed in the following table.

	Moments of Inertia ($kg \cdot m^2$)			*Products of Inertia* ($kg \cdot m^2$)		
	I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{yz}	I_{zx}
Rod AB	8/3	2/3	2	0	−1	0
Rod BC	4/3	0	4/3	0	0	0
Rod OD	0	2/3	2/3	0	0	0
Sum	4	4/3	4	0	−1	0

The moments and products of inertia of the rigid body are the sums of the corresponding inertia terms of the rods. Accordingly, the matrix of inertia of the rigid body is

$$[I_M] = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4/3 & 1 \\ 0 & 1 & 4 \end{bmatrix} \text{kg} \cdot \text{m}^2$$

By the commands

```
>> IM_matrix = [4 0 0; 0 4/3 1; 0 1 4];
>> prinertia(IM_matrix)
```

the principal moments of inertia and the direction cosines defined in Eq. (9.135) are obtained as follows

```
Computed Results
----------------
 Principal Moments of Inertia:
   I1 = 1
   I2 = 4
   I3 = 4.3333
```

```
 Principal Axes x', y' and z':
(direction cosines between xyz and x'y'z')
          x'      y'      z'
     ----------------------
x          0       1       0
y    -0.94868      0   0.31623
z     0.31623      0   0.94868
```

With the direction cosines, the orientations of the principal axes of inertia are completely determined. Figure 9.4.9 shows the principal axes that are plotted by function `prinertia`. Furthermore, for computation of the inertia parameters in $Ox'y'z'$, the coordinate transformation matrix in Eq. (9.134) has the form

$$[D] = \begin{bmatrix} \cos 25^\circ & -\sin 25^\circ & 0 \\ \sin 25^\circ & \cos 25^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Thus,

```
>> ang=25/180*pi;
>> D = [cos(ang)  -sin(ang)  0;  sin(ang)  cos(ang)  0; 0 0 1];
>> inertiaTF(IM_matrix, D)
```

yields

```
Computed inertia properties in frame x'y'z'
  Ix'x' = 3.5237,  Iy'y' = 1.8096,  Iz'z' = 4
  Ix'y' = 1.0214,  Iy'z' = -0.90631,  Iz'x' = -0.42262
```

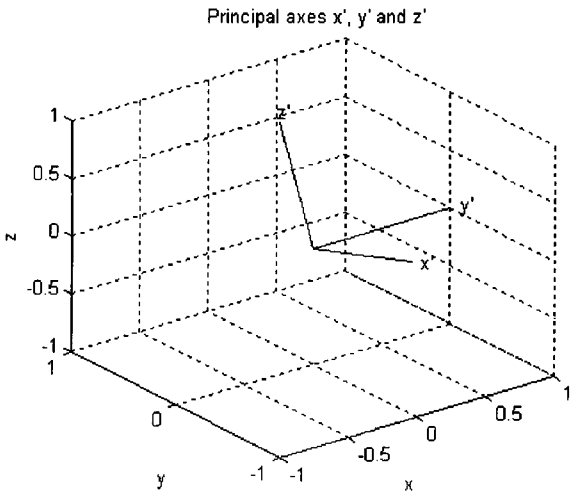


FIGURE 9.4.9

9.4.3 Energy and Momentum

The kinetic energy of a rigid body can be expressed by

$$T(t) = \frac{1}{2}mv_G^2 + \frac{1}{2}\boldsymbol{\omega} \cdot \boldsymbol{H}_G \tag{9.141}$$

where v_G is magnitude of the velocity of the mass center G, $\boldsymbol{\omega}$ is the angular velocity of the body, and $\boldsymbol{H}_G$ is the angular momentum of the body about G. According to Eqs. (9.123) and (9.124), the kinetic energy is written in the scalar form

$$\begin{aligned} T(t) &= \frac{1}{2}mv_G^2 + \frac{1}{2}\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}^T \begin{bmatrix} \bar{I}_{xx} & -\bar{I}_{xy} & -\bar{I}_{xz} \\ -\bar{I}_{yx} & \bar{I}_{yy} & -\bar{I}_{yz} \\ -\bar{I}_{zx} & -\bar{I}_{zy} & \bar{I}_{zz} \end{bmatrix} \begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix} \\ &= \frac{1}{2}mv_G^2 + \frac{1}{2}\left(\bar{I}_{xx}\omega_x^2 + \bar{I}_{yy}\omega_y^2 + \bar{I}_{zz}\omega_z^2 - 2\bar{I}_{xy}\omega_x\omega_y - 2\bar{I}_{yz}\omega_y\omega_z - 2\bar{I}_{zx}\omega_z\omega_x\right) \end{aligned} \tag{9.142}$$

where $\boldsymbol{\omega} = \omega_x\boldsymbol{i} + \omega_y\boldsymbol{j} + \omega_z\boldsymbol{k}$, and $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$ are the unit vectors of a frame $Gxyz$ centered at G and with fixed orientation. If the axes of $Gxyz$ are also the principal axes of inertia,

$$T(t) = \frac{1}{2}mv_G^2 + \frac{1}{2}\left(\bar{I}_{xx}\omega_x^2 + \bar{I}_{yy}\omega_y^2 + \bar{I}_{zz}\omega_z^2\right) \tag{9.143}$$

For a rigid body rotating about a fixed point O, its kinetic energy is

$$T(t) = \frac{1}{2}\boldsymbol{\omega} \cdot \boldsymbol{H}_O \tag{9.144}$$

which can be computed according to Eqs. (9.129) and (9.130).

The work-energy principle for a rigid body in three-dimensional motion is

$$W_{12} = \Delta T = T_2 - T_1 \tag{9.145}$$

where W_{12} is the work done by the external forces acting on the body. If all the external forces are conservative, the mechanical energy of the body remains constant, namely,

$$T_1 + V_1 = T_2 + V_2 \tag{9.146}$$

where V is the potential energy of the body. The linear momentum and angular momentum of a rigid body have been studied in Section 9.3.3. The impulse-momentum principles for three-dimensional motion have similar forms as those for plane motion. The linear impulse-momentum equation for a rigid body is

$$\int_{t_1}^{t_2} \sum \boldsymbol{F}\, dt = \boldsymbol{L}(t_2) - \boldsymbol{L}(t_1) \tag{9.147}$$

The angular impulse-momentum equation for a rigid body is

$$\int_{t_1}^{t_2} \sum \boldsymbol{M}_A dt = \boldsymbol{H}_A(t_2) - \boldsymbol{H}_A(t_1) \tag{9.148}$$

where A is either the mass center G of the body or a fixed point about which the body rotates.

9.4.4 Equations of Motion

Fundamental Equations The motion of a rigid body in three dimensions can be described by the force equation

$$\sum \boldsymbol{F} = m\boldsymbol{a}_G \tag{9.149}$$

and the moment equation

$$\sum \boldsymbol{M}_G = \dot{\boldsymbol{H}}_G \tag{9.150}$$

where $\sum \boldsymbol{F}$ is the resultant force, m is the mass of the body, $\boldsymbol{a}_G$ is the acceleration of the mass center G, $\sum \boldsymbol{M}_G$ is the resultant moment about G, and $\boldsymbol{H}_G$ is the angular momentum of the body about G. If a rigid body rotates about a fixed point O, its motion can be described by

$$\sum \boldsymbol{M}_O = \dot{\boldsymbol{H}}_O \tag{9.151}$$

where $\sum \boldsymbol{M}_O$ is the resultant moment about O, and $\boldsymbol{H}_O$ is the angular momentum of the body about O.

Moment Equation in a Rotating Frame In many problems, it is convenient to describe the angular momentum of a rigid body in a moving frame. Consider a rotating frame $Axyz$ with angular velocity $\boldsymbol{\Omega}$ in Fig. 9.4.10, where A is either the mass center of the body or a fixed point. Let the angular velocity of the body be $\boldsymbol{\omega}$. The rate of change of $\boldsymbol{H}_A$, according to Eq. (9.114), is

$$(\boldsymbol{H}_A)_{XYZ} = \frac{d}{dt}(\boldsymbol{H}_A)_{Gxyz} + \boldsymbol{\Omega} \times \boldsymbol{H}_A \tag{9.152}$$

Thus, the moment equation of the rigid body associated with the rotating frame is

$$\sum \boldsymbol{M}_A = \frac{d}{dt}(\boldsymbol{H}_A)_{Gxyz} + \boldsymbol{\Omega} \times \boldsymbol{H}_A \tag{9.153}$$

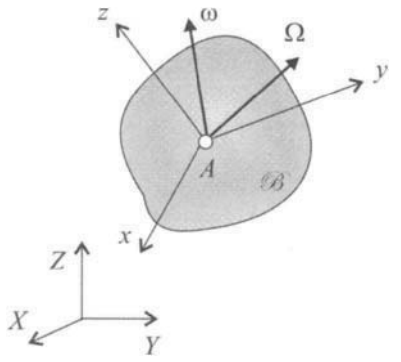


FIGURE 9.4.10 A body with a rotating frame.

Moment Equation in a Body Frame If the moving frame $Axyz$ in Fig. 9.4.10 is attached to the body such that $\boldsymbol{\Omega} = \boldsymbol{\omega}$, the frame is called a *body frame*. The moment equation associated with the body frame, by Eq. (9.153), is

$$\sum \boldsymbol{M}_A = \frac{d}{dt}(\boldsymbol{H}_A)_{Axyz} + \boldsymbol{\omega} \times \boldsymbol{H}_A \tag{9.154}$$

In the body frame, the moments and products of inertia of the body remain constant, and the scalar components of Eq. (9.154) are

$$\begin{aligned}
\sum M_x &= I_{xx}\dot{\omega}_x - (I_{yy} - I_{zz})\omega_y\omega_z + I_{xy}(\omega_z\omega_x - \dot{\omega}_y) - I_{zx}(\omega_x\omega_y + \dot{\omega}_z) - I_{yz}(\omega_y^2 - \omega_z^2) \\
\sum M_y &= I_{yy}\dot{\omega}_{yy} - (I_{zz} - I_{xx})\omega_z\omega_x + I_{yz}(\omega_x\omega_y - \dot{\omega}_z) - I_{xy}(\omega_y\omega_z + \dot{\omega}_x) - I_{zx}(\omega_z^2 - \omega_x^2) \\
\sum M_z &= I_{zz}\dot{\omega}_z - (I_{xx} - I_{yy})\omega_x\omega_y + I_{zx}(\omega_y\omega_z - \dot{\omega}_x) - I_{yz}(\omega_z\omega_x + \dot{\omega}_y) - I_{xy}(\omega_x^2 - \omega_y^2)
\end{aligned} \tag{9.155}$$

where $I_{xx}, I_{xy}, \ldots$ are the moments and products of inertia about A.

Euler's Equations When the axes of a body frame coincide with the principal axes of inertia of a rigid body, the products of inertia become zero, and the components of the angular momentum of the body become

$$H_x = I_{xx}\omega_x, \quad H_y = I_{yy}\omega_y, \quad H_z = I_{zz}\omega_z \tag{9.156}$$

The moment equations (9.155) thus reduce to

$$\begin{aligned}
\sum M_x &= I_{xx}\dot{\omega}_x - (I_{yy} - I_{zz})\omega_y\omega_z \\
\sum M_y &= I_{yy}\dot{\omega}_{yy} - (I_{zz} - I_{xx})\omega_z\omega_x \\
\sum M_z &= I_{zz}\dot{\omega}_z - (I_{xx} - I_{yy})\omega_x\omega_y
\end{aligned} \tag{9.157}$$

which are known as Euler's equations. Euler's equations, along with the scalar components of the force equation (9.149)

$$\sum F_x = m\bar{a}_x, \quad \sum F_y = m\bar{a}_y, \quad \sum F_z = m\bar{a}_z \tag{9.158}$$

form a set of six differential equations that is commonly used in rigid body dynamics. These equations can be solved via numerical integration.

Rotation about a Fixed Axis A rigid body rotates about a fixed axis A-A with rotation speed ω; see Fig. 9.4.11 where $OXYZ$ is a fixed frame and $Oxyz$ is a body frame that has the same angular velocity as the rigid body ($\boldsymbol{\Omega} = \boldsymbol{\omega} = \omega\boldsymbol{k}$). Because, $\omega_x = 0$, $\omega_y = 0$, and $\omega_z = \omega$, the moment equations (9.155) becomes

$$\begin{aligned}
\sum M_x &= -I_{zx}\alpha + I_{yz}\omega^2 \\
\sum M_y &= -I_{yz}\alpha - I_{zx}\omega^2 \\
\sum M_z &= I_{zz}\alpha
\end{aligned} \tag{9.159}$$

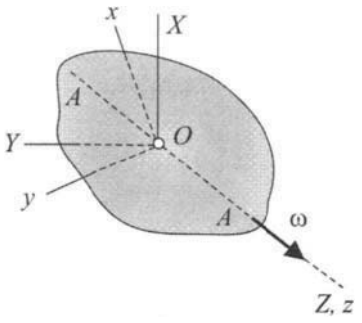


FIGURE 9.4.11 A body rotating about a fixed axis.

where $\alpha = \dot{\omega}$ is the angular acceleration of the body, and $I_{zz}, I_{yz} \ldots$ are the moment and products of inertia of the body about O. Equations (9.158) and (9.159) are useful for studying the balancing and support reactions of rotating machinery.

EXAMPLE 4.3

In Fig. 9.4.12, a spinning disk of mass m and radius r with a shaft of length $|AB| = 2L$ is mounted on a rotating platform of constant speed ω_0. The disk rotates at a constant speed ω_d relative to the platform. Assume that the mass of the shaft is negligible. Determine the bearing reactions at A and B. Assume that bearing B restrains the shaft in the longitudinal direction.

Solution. Let $OXYZ$ be a fixed frame with unit vectors $\boldsymbol{I}, \boldsymbol{J}$, and $\boldsymbol{K}$. Let $Gxyz$ be a body frame that is attached to the disk at the mass center G, with unit vectors $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$. Assume that plane Gyz coincides with plane OYZ at the moment. The angular velocity and acceleration of the disk are

$$\boldsymbol{\omega} = \omega_0 \boldsymbol{K} + \omega_d \boldsymbol{j} = (\omega_d - \omega_0 \cos\beta)\boldsymbol{j} + \omega_0 \sin\beta\, \boldsymbol{k}$$

$$\dot{\boldsymbol{\omega}} = \dot{\omega}_0 \boldsymbol{K} + \dot{\omega}_d \boldsymbol{j} + \boldsymbol{\omega} \times \omega_d \boldsymbol{j} = -\omega_0 \omega_d \sin\beta\, \boldsymbol{i}$$

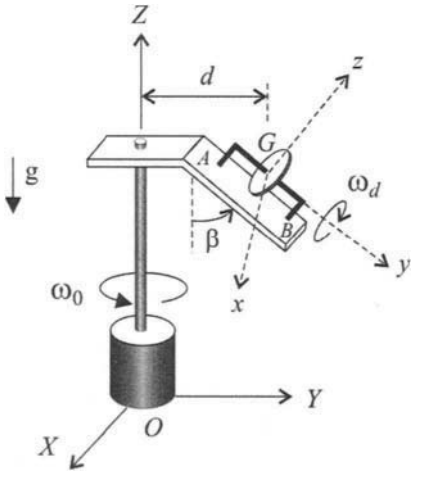


FIGURE 9.4.12

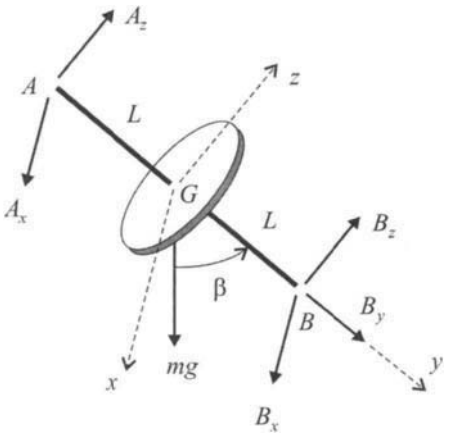


FIGURE 9.4.13

So, the components of the angular velocity and acceleration are

$$\omega_x = 0, \quad \omega_y = \omega_d - \omega_0 \cos\beta, \quad \omega_z = \omega_0 \sin\beta$$
$$\dot{\omega}_x = -\omega_0 \omega_d \sin\beta, \quad \dot{\omega}_y = 0, \quad \dot{\omega}_z = 0$$

By the free-body diagram in Fig. 9.4.13, the Euler's equations for the disk are

$$\sum M_x = (B_z - A_z)\, L = -2I\omega_0\omega_d \sin\beta + I\omega_0^2 \cos\beta \sin\beta$$
$$\sum M_z = (A_x - B_x)\, L = 0$$

where $I = \frac{1}{4}mr^2$. The acceleration components of the mass center G are

$$\bar{a}_x = 0, \quad \bar{a}_y = -d\omega_0^2 \sin\beta, \quad \bar{a}_z = -d\omega_0^2 \cos\beta$$

Thus, the force equations of the disk are

$$\sum F_x = A_x + B_x = 0$$
$$\sum F_y = B_y + mg\cos\beta = -md\omega_0^2 \sin\beta$$
$$\sum F_z = A_z + B_z - mg\sin\beta = -md\omega_0^2 \cos\beta$$

From the above five equations, the reactions are determined as follows:

$$A_x = 0$$
$$B_x = 0$$
$$B_y = -m(g\cos\beta + d\omega_0^2 \sin\beta)$$
$$A_z = \frac{m}{2}\left(g\sin\beta - d\omega_0^2 \cos\beta\right) + \frac{mr^2\omega_0}{8L}(2\omega_d - \omega_0 \cos\beta)\sin\beta$$
$$B_z = \frac{m}{2}\left(g\sin\beta - d\omega_0^2 \cos\beta\right) - \frac{mr^2\omega_0}{8L}(2\omega_d - \omega_0 \cos\beta)\sin\beta$$

9.4.5 Rotation of Axisymmetric Bodies

The rotation of an axisymmetric rigid body about a fixed point or its mass center has important applications in gyroscopes, satellites, projectiles, and spacecraft. The three-dimensional motion of such a rigid body is often described by a set of new kinematics variables called *Euler angles*.

Euler Angles In Fig. 9.4.14, an axisymmetric rigid body rotates about point O, which is either a fixed point or the mass center of the body. Select a rotating frame $Oxyz$, such that the z-axis is the axis of symmetry of the body. In addition, the body spins about the z-axis.

The motion of the body is described in a special way. To begin, place the body in a fixed frame $OXYZ$, with the Z-axis coinciding with the axis of symmetry of the body. (The direction of the Z-axis can be chosen arbitrarily.) Then rotate the body in the following three steps:

(a) Rotate the body about the Z-axis through an angle ϕ, to obtain frame $Ox_1y_1z_1$ in Fig. 9.4.15(a). This process is called *precession*. The angle ϕ is called the *angle of precession*, and its time derivative $\dot{\phi}$ is the *rate of precession*.

(b) Rotate the body about the x_1-axis through an angle θ, to obtain frame $Oxyz$ in Fig. 9.4.15(b). This process is called *nutation*. The angle θ is called the *angle of nutation*, and its time derivative $\dot{\theta}$ is the *rate of nutation*.

(c) Rotate the body, an angle ψ about the z-axis; see Fig. 9.4.15(c). This process is called *spin*. The angle ψ is called the *angle of spin*, and its time derivative is the *rate of spin*.

As can be seen, the rotating frame $Oxyz$ is obtained through precession and nutation; the rotation of the body is completely described by three angles, ϕ, θ, and ψ, which are called *Euler angles*. The Z-axis is known as the *invariable line* while the x-axis is referred to as the *nodal line*.

According to the above discussion, the angular velocity of the rotating frame is $\boldsymbol{\Omega} = \dot{\boldsymbol{\phi}} + \dot{\boldsymbol{\theta}}$, and the angular velocity of the rigid body is $\boldsymbol{\omega} = \dot{\boldsymbol{\phi}} + \dot{\boldsymbol{\theta}} + \dot{\boldsymbol{\psi}}$. The components of these angular velocities relative to frame $Oxyz$ are

$$\begin{aligned} \Omega_x &= \dot{\theta}, \quad \Omega_y = \dot{\phi}\sin\theta, \quad \Omega_z = \dot{\phi}\cos\theta \\ \omega_x &= \dot{\theta}, \quad \omega_y = \dot{\phi}\sin\theta, \quad \omega_z = \dot{\phi}\cos\theta + \dot{\psi} \end{aligned} \tag{9.160}$$

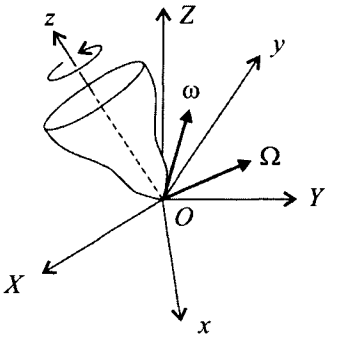
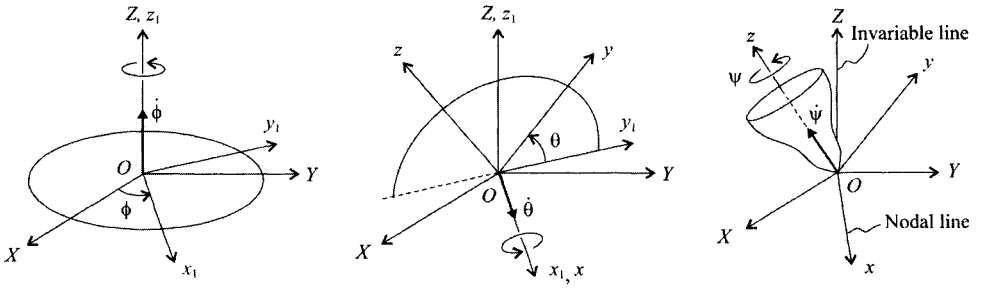


FIGURE 9.4.14 An axisymmetric body rotating about a fixed point or its mass center.



(a) Precession (b) Nutation (c) Spin

FIGURE 9.4.15 The rotation of an axisymmetric body in Euler angles.

Coordinate Transformation Let $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$ be the unit vectors of the rotating frame $Oxyz$. Let $\boldsymbol{I}, \boldsymbol{J}$, and $\boldsymbol{K}$ be the unit vectors of the fixed frame $OXYZ$. A position vector $\boldsymbol{r}$ can be expressed by

$$\boldsymbol{r} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k} = X\boldsymbol{I} + Y\boldsymbol{J} + Z\boldsymbol{K} \tag{9.161}$$

The coordinates in the fixed frame are related to those in the rotating frame by

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{bmatrix} \cos\phi & -\cos\theta\sin\phi & \sin\theta\sin\phi \\ \sin\phi & \cos\theta\cos\phi & -\sin\theta\cos\phi \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{9.162}$$

Moment Equations of Motion From Eq. (9.160), the angular velocity $\boldsymbol{\omega}$ of the body in general is different from the angular velocity $\boldsymbol{\Omega}$ of the frame $Oxyz$. So, the rotating frame is not a body frame, and Euler's equations (9.157) are not applicable here. Because the z-axis is the axis of symmetry of the body, the moments and products of inertia of the body can be written as

$$I = I_{xx} = I_{yy}, \quad I_z = I_{zz}, \quad I_{xy} = I_{yz} = I_{zx} = 0 \tag{9.163}$$

Substituting Eqs. (9.160) and (9.163) into Eq. (9.153) yields the moment equations of motion of the axisymmetric body

$$\begin{aligned} \sum M_x &= I\ddot{\theta} + (I_z - I)\dot{\phi}^2 \sin\theta\cos\theta + I_z\dot{\phi}\dot{\psi}\sin\theta \\ \sum M_y &= I\ddot{\phi}\sin\theta + 2I\dot{\theta}\dot{\phi}\cos\theta - I_z\dot{\theta}\left(\dot{\psi} + \dot{\phi}\cos\theta\right) \\ \sum M_z &= I_z\left(\ddot{\psi} + \ddot{\phi}\cos\theta - \dot{\theta}\dot{\phi}\sin\theta\right) \end{aligned} \tag{9.164}$$

Note that when $\sin\theta = 0$, the second equation of (9.164) becomes singular. Equations (9.164) can also be written as

$$\begin{aligned} \sum M_x &= I\ddot{\theta} + I_z\omega_z\dot{\phi}\sin\theta - I\dot{\phi}^2\sin\theta\cos\theta \\ \sum M_y &= I\ddot{\phi}\sin\theta + 2I\dot{\theta}\dot{\phi}\cos\theta - I_z\dot{\theta}\omega_z \\ \sum M_z &= I_z\dot{\omega}_z \end{aligned} \tag{9.165}$$

where $\omega_z = \dot{\phi}\cos\theta + \dot{\psi}$ is from Eq. (9.160).

Steady Precession Steady precession of an axisymmetric rigid body occurs when $\theta, \dot{\phi}$, and $\dot{\psi}$ are constant. The motion of the body in this case is governed by

$$\begin{aligned} \sum M_x &= (I_z - I)\dot{\phi}^2 \sin\theta\cos\theta + I_z\dot{\phi}\dot{\psi}\sin\theta \\ \sum M_y &= 0, \quad \sum M_z = 0 \end{aligned} \tag{9.166}$$

The above equations indicate that in a steady precession, the body is subject to constant resultant moment about the x-axis, and no resultant moments about the y- and z-axes.

Torque-Free Motion If the resultant moment of the external forces about the mass center G of a rigid body is zero, the angular momentum $\boldsymbol{H}_G$ of the body about G, by Eq. (9.150), is a constant vector. Let $GXYZ$ be a fixed frame with the Z-axis in the direction of $\boldsymbol{H}_G$; see Fig. 9.4.16. The components of $\boldsymbol{H}_G$ relative to $Gxyz$ are

$$H_x = 0, \quad H_y = H_G\sin\theta, \quad H_z = H_G\cos\theta \tag{9.167}$$

where H_G is the magnitude of $\boldsymbol{H}_G$. Because the xyz axes are the principal axes of inertia,

$$H_x = \bar{I}\omega_x, \quad H_y = \bar{I}\omega_y, \quad H_z = \bar{I}_z\omega_z \tag{9.168}$$

It follows from Eqs. (9.167), (9.168), and (9.160) that

$$\omega_x = \dot{\theta} = 0, \quad \omega_y = \frac{H_G}{\bar{I}}\sin\theta, \quad \omega_z = \frac{H_G}{\bar{I}_z}\cos\theta \tag{9.169}$$

Note that a torque-free motion is a steady precession because θ, $\dot{\phi}$, and $\dot{\psi}$ are constant.

So, in a torque-free motion, the nutation angle θ is a constant, the angular velocity $\boldsymbol{\omega}$ of the body stays in the plane Gyz, and the precession velocity $\dot{\phi}$ and spin velocity $\dot{\psi}$ are constant. In other words,

$$\theta = \theta_0, \quad \dot{\phi} = \frac{H_G}{\bar{I}}, \quad \dot{\psi} = \left(\frac{1}{\bar{I}_z} - \frac{1}{\bar{I}}\right)H_G\cos\theta_0, \tag{9.170}$$

where Eqs. (9.160) have been used. Furthermore, let the angle between the angular velocity $\boldsymbol{\omega}$ and the z-axis be α. The angles θ and α are related by

$$\tan\alpha = \frac{\omega_y}{\omega_z} = \mu\tan\theta, \quad \text{with } \mu = \frac{\bar{I}_z}{\bar{I}} \tag{9.171}$$

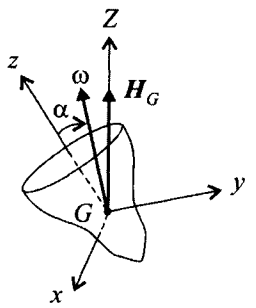


FIGURE 9.4.16 An axisymmetric body in torque-free motion.

When $\mu < 1$ ($\bar{I}_z < \bar{I}$), which is a case of an elongated body along the z-axis, the motion is called *direct* or *regular precession*. When $\mu > 1$ ($\bar{I}_z > \bar{I}$), which is the case of a flattened body, the motion is called *retrograde precession*.

Numerical Simulation and Animation The motion of an axisymmetric rigid body can be determined by numerical integration. To this end, define the state vector

$$\eta(t) = \begin{pmatrix} \phi(t) & \theta(t) & \psi(t) & \dot{\phi}(t) & \dot{\theta}(t) & \dot{\psi}(t) \end{pmatrix}^T \tag{9.172}$$

where ϕ, θ, and ψ are the precession, nutation, and spin angles of the body, respectively. Equations (9.164) are rewritten in terms of $\eta(t)$

$$\dot{\eta}(t) = \Phi(t, \eta(t)); \quad \eta(0) = \eta_0 \tag{9.173}$$

where

$$\begin{aligned} \eta_0 &= \begin{pmatrix} \phi(0) & \theta(0) & \psi(0) & \dot{\phi}(0) & \dot{\theta}(0) & \dot{\psi}(0) \end{pmatrix}^T \\ \Phi(t, \eta(t)) &= \begin{pmatrix} \dot{\phi}(t) & \dot{\theta}(t) & \dot{\psi}(t) & \pi_1 & \pi_2 & \pi_3 \end{pmatrix}^T \end{aligned} \tag{9.174}$$

with

$$\begin{aligned} \pi_1 &= \frac{1}{I \sin\theta} \left\{ \sum M_y - 2I\dot{\theta}\dot{\phi}\cos\theta + I_z\dot{\theta}\left(\dot{\psi} + \dot{\phi}\cos\theta\right) \right\} \\ \pi_2 &= \frac{1}{I} \left\{ \sum M_x - (I_z - I)\dot{\phi}^2 \sin\theta\cos\theta - I_z\dot{\phi}\dot{\psi}\sin\theta \right\} \\ \pi_3 &= -\frac{\cot\theta}{I} \left\{ \sum M_y - 2I\dot{\theta}\dot{\phi}\cos\theta + I_z\dot{\theta}\left(\dot{\psi} + \dot{\phi}\cos\theta\right) \right\} + \frac{1}{I_z} \left\{ \sum M_z + I_z\dot{\theta}\dot{\phi}\sin\theta \right\} \end{aligned} \tag{9.175}$$

Developed based on the Runge-Kutta method given in Eq. (9.36), a MATLAB function `axisym3d` is available for the solution of the state equation (9.173); see Window 4.3. Also, function `eulermotion` is available for animation of the motion of an axisymmetric body; see Window 4.4.

Window 4.3. Function `axisym3d`

MATLAB Function: **`axisym3d`**

Purpose:
To plot the Euler angles and rotation velocities of an axisymmetric rigid body by Runge-Kutta numerical integration of the state equation (9.173).

Synopsis:

```
axisym3d(Iz, Ix, 'ForceFn', eta0)
axisym3d(Iz, Ix, 'ForceFn', eta0, tf, npts)
[t, eta] = axisym3d(Iz, Ix, 'ForceFn', eta0, tf, npts)
```

Window 4.3. Function axisym3d (*Continued*)

Description:
`axisym3d(Iz, Ix, 'ForceFn', eta0)` plots the time histories of the Euler angles and precession, nutation, and spin velocities of the body. Here, `Iz` and `Ix` are the moments of inertia of the body about the z-axis and x-axis, respectively, `ForceFn` is the name of a user-provided M-file function that describes the external moments in Eq. (9.164), and `eta0` is the initial state vector given in Eq. (9.174). The user-provided function has the format

```
y = ForceFn(t, z)
```

where `t` is the time; `z` is the state vector $\eta(t)$ defined by Eq. (9.172); and `y` is the vector of external moments, with `y(1)` being $\sum M_x$, `y(2)` $\sum M_y$, and `y(3)` $\sum M_z$. If no external force acts on the body, simply use

```
axisym3d(Iz, Ix, 0, eta0) or axisym3d(Iz, Ix, [], eta0).
```

`axisym3d(Iz, Ix, 'ForceFn', eta0, tf, npts)` plots the dynamic response of the rigid body with specified parameters `tf` and `npts`, where `tf` is total computation time, and `npts` is the number of time points in the region $0 \leq t \leq$ `tf`. The step size h of the Runge-Kutta algorithm is controlled by $h =$ `tf/(npts` $- 1$). By default, `tf` $= 5$ and `npts` $= 1{,}001$. If there is no external moment, `'ForceFn'` is replaced by 0 or [ ]. If the body has zero initial disturbances, `eta0` is replaced by 0 or [ ].

`[t, eta] = axisym3d(Iz, Ix, 'ForceFn', eta0, tf, npts)` returns the times of simulation in a vector `t`, and the time history of the state vector in a six-row matrix `eta`. The first three rows of `eta` contain the data of Euler angles ϕ, θ, and ψ while the last three rows their time derivatives $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$. With `t` and `eta` the motion of the rigid body can be plotted by the MATLAB function `plot`. For instance, `plot(t, eta (1,:), t, eta (4,:))`, plots the precession angle ϕ and precession rate $\dot{\phi}$ in one figure.

Note: When $\sin\theta = 0$, `axisym3d` may not work well, due to the singularity of Eqs. (9.164).

Window 4.4. Function eulermotion

MATLAB Function: **`eulermotion`**

Purpose:
To animate the motion of an axisymmetric rigid body in a fixed frame.

Synopsis:

```
eulermotion(t, angles)
eulermotion(t, angles, n_frames, IS_control)
[F, t_frame] = eulermotion(t, angles, n_frames, IS_control)
```

Window 4.4. Function `eulermotion` (*Continued*)

Description:

`eulermotion(t, angles)` plays the animated motion of the rigid body, where `t` is a vector of times, and `angles` is a matrix with the first row `angles(1,:)` containing the time history of the precession angle ϕ, the second row `angles(2,:)` the nutation angle θ, and the third row `angles(3,:)` the spin angle ψ. The row length of `angles` must be in agreement with that of vector `t`. The matrix `angles` may have more than three rows, as long as the first three rows are for the Euler angles.

`eulermotion(t, angles, n_frames, IS_control)` plays the animated motion of the rigid body, with `n_frames` being the total number of frames in the time period spanned by vector `t`, and `IS_control` an animation control parameter with the options

`IS_control = 0`	play frames continuously
`IS_control = 1`	play frames one by one

By default, `IS_control = 0` and `n_frames = 101`.

`[F, t_frame] = eulermotion(t, angles, n_frames, IS_control)` returns a set of frames of the animated motion in a matrix `F`, which later on can be played by the MATLAB function `movie(F)`, and the times of animation in a vector `t_frame`.

In animation by function `eulermotion`, a rotating L-shaped bar *OAB* shown in Fig. 9.4.17 is used to visualize the motion of an axisymmetric rigid body. The segment *OA* illustrates the motion (precession and nutation) of the axis of symmetry of the body relative to the fixed frame *OXYZ*; the segment *AB* shows the spin motion of the body relative to its axis of symmetry (the z-axis).

The input arguments `t` and `angles` of function `eulermotion` can be obtained analytically if steady precession or torque-free motion is concerned; see Example 4.4. For general motion of an axisymmetric rigid body, however, Eqs. (9.164) can be solved numerically by function `axisym3d`. In this case, the commands

```
[t, eta] = axisym3d(Iz, Ix, 'ForceFn', eta0);
eulermotion(t, eta)
```

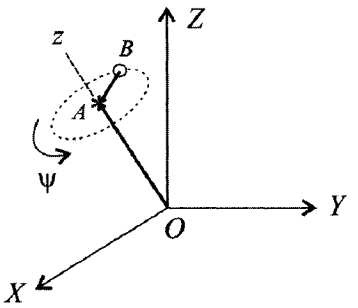


FIGURE 9.4.17 Visualization of the motion of an axisymmetric body.

conveniently animate the motion of the body. Also, the software MATLAB has a function ode45 for numerical solutions of differential equations including Eqs. (9.164) and (9.165).

EXAMPLE 4.4

In Fig. 9.4.18, a top with mass m = 5 kg and moments of inertia $I = I_{xx} = I_{yy} = 0.01$ kg · m^2 and $I_z = I_{zz} = 0.025$ kg · m^2 is supported at the fixed point O. The top is in steady precession with the rate of spin $\dot{\psi} = 18.7$ rad/s and the rate of precession $\dot{\phi} = 2\pi$ rad/s. The distance between the fixed point and mass center is $c = |OG| = 0.07$ m. Determine nutation angle θ and animate the motion of the rigid body.

Solution. By Eqs. (9.166),

$$M_x = mgc \sin\theta = (I_z - I)\dot{\phi}^2 \sin\theta \cos\theta + I_z \dot{\phi}\dot{\psi} \sin\theta$$

which leads to

$$\cos\theta = \frac{mgc - I_z \dot{\phi}\dot{\psi}}{(I_z - I)\dot{\phi}^2}$$

Substituting the values of the parameters into the above expression yields

$$\theta = 0.5776 \text{ rad } (33.1^\circ)$$

The Euler angles of the top can be expressed as

$$\phi(t) = 2\pi t, \quad \theta(t) \equiv 0.5776, \quad \psi(t) = 18.7t \text{ (all in rad)}$$

The motion of the top in a time period of five seconds is animated by

```
>> tt = linspace(0,5,1001);
>> ang_precession = 2*pi*tt;
>> ang_nutation = 0.5776*ones(1,1001);
>> ang_spin = 18.7*tt;
>> angles = [ang_precession; ang_nutation; ang_spin];
>> eulermotion(tt, angles)
```

Shown in Fig. 9.4.19 are four of the frames obtained from the animation.

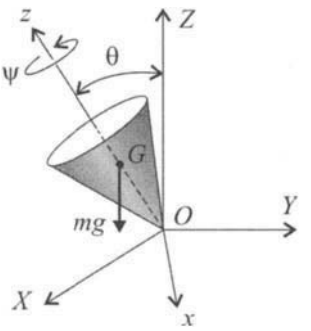


FIGURE 9.4.18

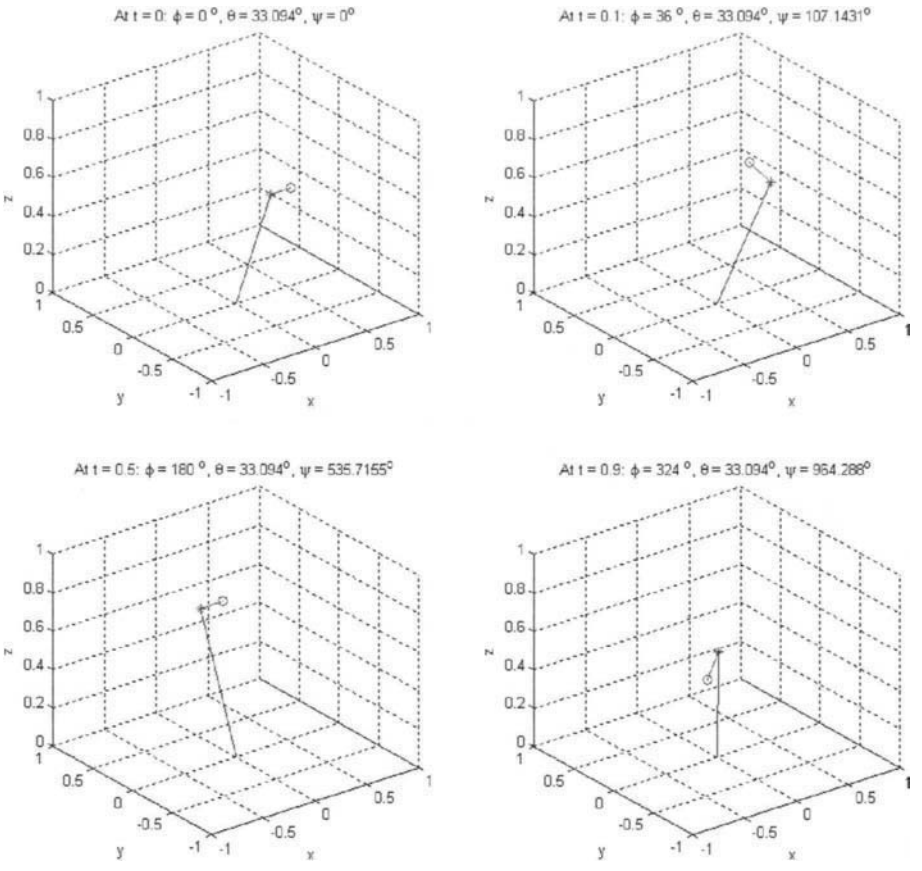


FIGURE 9.4.19

EXAMPLE 4.5

Consider the same rigid body in Example 4.4, with the initial conditions

$$\phi(0) = 0, \quad \theta(0) \equiv 0.5776 \text{ rad}, \quad \psi(t) = 0$$

$$\dot{\phi}(0) = 2\pi \text{ rad/s}, \quad \dot{\theta}(0) \equiv 0, \quad \dot{\psi}(t) = 18.7 \text{ rad/s}$$

The external moments applied to the body are

$$\sum M_x = mgc \sin\theta - 0.025\dot{\theta}$$

$$\sum M_y = \begin{cases} -4.8, & 1\text{ s} \le t < 1.05\text{ s} \\ 0, & \text{otherwise} \end{cases}$$

$$\sum M_z = 0$$

all of which are in N·m. Here $-0.025\dot{\theta}$ is a damping torque against the nutation of the body, and $\sum M_y$ describes an impulse moment acting between time $t = 1$ s and time $t = 1.05$ s. The motion of the body is computed by the following commands

```
>> Ix = 0.01; Iz = 0.025;
>> eta0 = [0; 0.5776; 0; 2*pi; 0; 18.7];
>> axisym3d(Iz, Ix, 'force_exm45', eta0)
```

where the user-provided M-file function `force_exm45` is of the form

```
% M-file named force_exm45
function y = force_exm45(tt, a)
m = 5; c = 0.07; g = 9.81;
nu_angle = a(2); nu_rate = a(5);
Mx = m*g*c*sin(nu_angle)-0.025*nu_rate;
My = 0;
if tt >= 1 & tt < 1.05
  My = -4.8;
end
y = [Mx; My; 0];
```

See Fig. 9.4.20 for the time histories of the Euler angles and angular velocities of the body. The body is in a steady precession during the first second. This is because the initial conditions match those of the steady precession given in Example 4.4. The action of the impulse moment that takes place at $t = 1$ s destroys the steady motion. The transients of the body excited by the impulse moment are then being settled down by the damping torque through nutation. This eventually leads to a new steady precession with a different set of constant nutation angle, precession rate, and spin velocity: $\theta = 0.9539$ rad, $\dot{\phi} = 8.2$ rad/s, $\dot{\psi} = 13.9024$ rad/s. This switch from one steady precession to another can also be seen through animation by `eulermotion`.

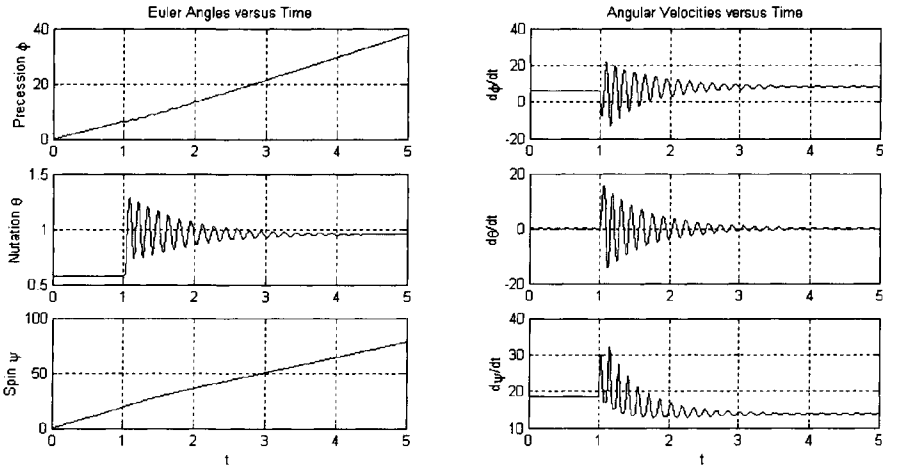


FIGURE 9.4.20

9.5 Quick Solution Guide

Equation of Motion

Single particle

$$ma_x = \sum F_x, \quad ma_y = \sum F_y, \quad ma_z = \sum F_z$$

System of particles

$$\sum_i \boldsymbol{F}_i = m\boldsymbol{a}_G, \quad \sum_i \boldsymbol{M}_{i,G} = \dot{\boldsymbol{H}}_G$$

where $m = \sum_i m_i$ is total mass; $\boldsymbol{a}_G = \frac{1}{m}\sum_i m_i \boldsymbol{a}_i$ is the acceleration of mass center G; and $\boldsymbol{H}_G = \sum_i \boldsymbol{\rho}_i \times m_i \boldsymbol{v}_i$ is the angular momentum about G.

Rigid Bodies in Two Dimensions

$$\sum F_x = m\bar{a}_x, \quad \sum F_y = m\bar{a}_y, \quad \sum M_G = \bar{I}_z \alpha$$

Rigid Bodies in Three Dimensions

$$\sum \boldsymbol{F} = m\boldsymbol{a}_G, \quad \sum \boldsymbol{M}_G = \dot{\boldsymbol{H}}_G$$

where $\boldsymbol{a}_G$ is the acceleration of mass center G, and $\boldsymbol{H}_G(t) = \int_{\mathscr{B}} \boldsymbol{r}' \times \boldsymbol{v} dm$ is the angular momentum about G.

Euler's Equations

$$\sum M_x = I_{xx}\dot{\omega}_x - (I_{yy} - I_{zz})\omega_y\omega_z$$
$$\sum M_y = I_{yy}\dot{\omega}_{yy} - (I_{zz} - I_{xx})\omega_z\omega_x$$
$$\sum M_z = I_{zz}\dot{\omega}_z - (I_{xx} - I_{yy})\omega_x\omega_y$$

which are for the rotation of a rigid body in three dimensions when its principal axes of inertia coincide with the axes of a body frame.

MATLAB Functions

This chapter contains a toolbox of MATLAB functions for determination of motion and for computation of inertia parameters in particle and rigid-body dynamics; see the table below. Refer to the corresponding window or section for the utility of each function.

Dynamics of Particles

`resultant`	Compute resultant force and moment of a system of forces	Window 2.1
`part2d`	Simulate the motion for a particle in two dimensions	Window 2.2
`part3d`	Simulate the motion for a particle in three dimensions	Window 2.2
`partmotion`	Animate the motion of a particle in two or three dimensions	Window 2.3
`mscenter`	Compute total mass and mass center of a particle system	Window 2.4

(continued)

Rigid Body Dynamics in Two Dimensions		
massmin	Compute mass moments of inertia of a rigid body	Window 3.1
rigid2d	Plot dynamic response of a rigid body in plane motion	Window 3.2
animotion2d	Animate the two-dimensional motion of a rigid body	Window 3.3
Rigid Body Dynamics in Three Dimensions		
inertiaTF	Transform the moments and products of inertia from one coordinate system to another	Window 4.1
prinertia	Determine the principal axes and principal moments of inertia of a rigid body	Window 4.2
axisym3d	Plot Euler angles and angular velocities of an axisymmetric rigid body	Window 4.3
eulermotion	Animate the motion of an axisymmetric rigid body by using Euler angles	Window 4.4
Utilities		
TBdemo	Show how the Toolbox works	Section 9.1
TBinfo	List MATLAB functions contained in this Toolbox	Section 9.1
RunEx	Run all the numerical examples contained in this chapter	Section 9.1

9.6 References

1. Beer F P, Johnston E R, Jr. 1996 *Vector Mechanics for Engineers*, 6th ed., McGraw-Hill, Inc.: New York.
2. Greenwood D T 1977 *Classical Dynamics*, Prentice-Hall, Inc.: Englewood Cliffs, New Jersey.
3. Goldstein H 1980 *Classical Mechanics*, 2nd ed., Addison-Wesley Publishing Co.: Menlo Park, California.
4. McGill D J, King W W 1989 *Engineering Mechanics—An Introduction to Dynamics*, 2nd ed., PWS-KENT Publishing Co.: Boston.
5. Meriam J L, Kraige, L G 1997 *Engineering Mechanics—Dynamics*, 4th ed., John Wiley & Sons, Inc.: New York.
6. Pytel A, Kiusalaas J 1999 *Engineering Mechanics—Dynamics*, 2nd ed., Brooks/Cole Publishing Co.: New York.
7. Shames I H 1997 *Engineering Mechanics—Statics and Dynamics*, 4th ed., Prentice-Hall, Inc.: Upper Saddle River, New Jersey.

10 Vibration Analysis of One-Degree-of-Freedom Systems

Inside
- Getting Started
- System Description
- Time Response
- Analytical Vibration Solutions
- Frequency Response
- Response to Periodic Excitation
- Nonlinear Vibration
- Quick Solution Guide
- References

10.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and computer software (MATLAB functions) for vibration analysis of dynamic systems with one degree of freedom (1-DOF).

System Requirements for the MATLAB Toolbox

- PC with Win 98/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up)

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

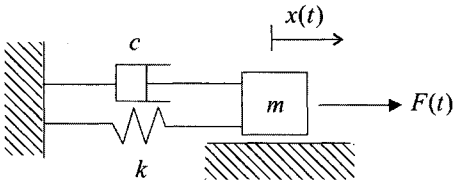


FIGURE 10.1.1

(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

Quick Tutorial

To show how to use the Toolbox, consider a spring-mass-damper system in Fig. 10.1.1, with mass $m = 1$, damping coefficient $c = 1$, and spring coefficient $k = 4$. Assume that the system is subject to the initial displacement $x(0) = 0.2$, and initial velocity $\dot{x}(0) = -2$, and a step (constant) external load $F(t) = 4$ for $t \geq 0$. The total response of the system under these excitations can be obtained by typing the following in the MATLAB command window:

```
>> m = 1; c = 1; k = 4;
>> setldof(m, c, k)
>> x0 = 0.2; v0 = -2;
>> load = [1 4];
>> y = freevib(x0, v0) + forcedvib(load);
>> plotvib(y)
```

where >> is the prompt in the MATLAB command window. This yields the time history of the displacement of the system in Fig. 10.1.2.

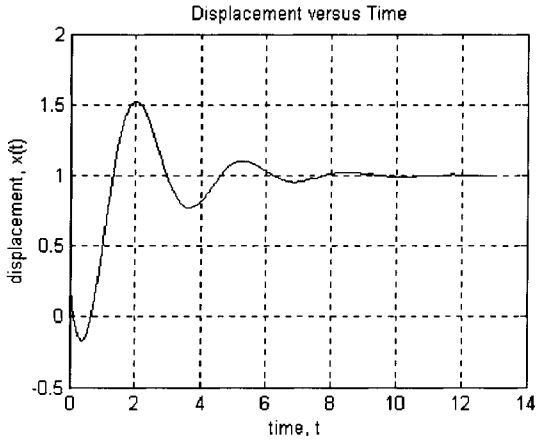


FIGURE 10.1.2

In the above vibration simulation, four functions from the Toolbox are called:

(a) Function **setldof** that inputs system parameters m, c, and k;
(b) Function **freevib** that computes free response subject to initial disturbances;
(c) Function **forcedvib** that computes forced response subject to an external force; and
(d) Function **plotvib** that plots system response versus time.

These functions, along with others, will be introduced in the subsequent sections.

For a quick solution, the reader may refer directly to the Quick Solution Guide (Section 10.8), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To better understand how the Toolbox works, the user is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 10.9.

10.2 System Description

This section presents the equation of motion, damping status, and characteristic roots for one-degree-of-freedom (1-DOF) vibrating systems. MATLAB functions from the Toolbox are also introduced for system description and vibration analysis.

Equation of Motion The displacement $x(t)$ of the spring-mass-damper system in Fig. 10.2.1 is governed by the differential equation (Reference 2 and 6)

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t) \tag{10.1}$$

where m is the mass (kg), c the coefficient of viscous damping (kg/s), k the spring coefficient (N/m), and $F(t)$ an external force. Equation (10.1) can be written as

$$\ddot{x}(t) + 2\xi\omega_n\dot{x}(t) + \omega_n^2 x(t) = f(t) = \frac{1}{m}F(t) \tag{10.2}$$

in which parameters ω_n and ξ are defined below:

$$\begin{aligned} &\text{Circular natural frequency} && \omega_n = \sqrt{k/m}, \quad \text{in rad/s} \\ &\text{Damping ratio} && \xi = \frac{c}{2\sqrt{mk}} = \frac{c}{c_{cr}} \end{aligned} \tag{10.3}$$

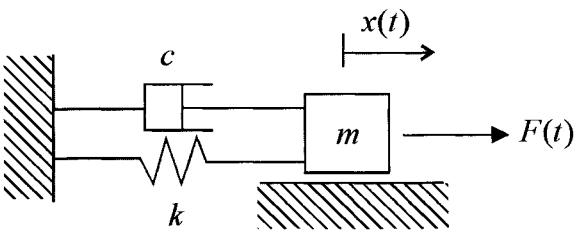


FIGURE 10.2.1 A spring-mass-damper system.

Here $c_{cr} = 2\sqrt{mk}$ is called *critical damping coefficient*. Additionally, the following two parameters are often used in vibration analysis:

$$\text{Natural frequency} \quad f_n = \frac{\omega_n}{2\pi} = \frac{1}{2\pi}\sqrt{k/m}, \quad \text{in Hz}$$

$$\text{Natural period} \quad T = \frac{1}{f_n} = \frac{2\pi}{\omega_n}, \quad \text{in seconds}$$

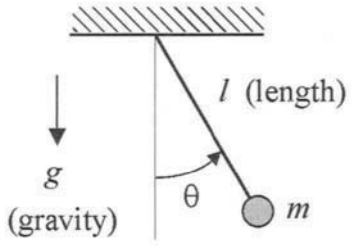
Although a spring-mass-damper system is considered, the equation of motion (10.1) is generic for 1-DOF linear vibrating systems. In general, a 1-DOF vibrating system is described by

$$\alpha_M \ddot{z}(t) + \alpha_C \dot{z}(t) + \alpha_K z(t) = F(t) \tag{10.4}$$

where $z(t)$ is a general coordinate, and α_M, α_C, and α_K are effective inertia, damping, and stiffness, respectively. Listed in Table 10.2.1 are some examples of 1-DOF vibrating systems, where damping elements and external forces are not included for simplicity. As can be seen,

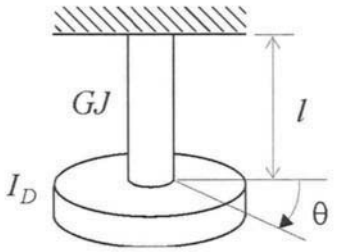
TABLE 10.2.1 Samples of 1-DOF Vibrating Systems (in Small Oscillations)

Simple pendulum



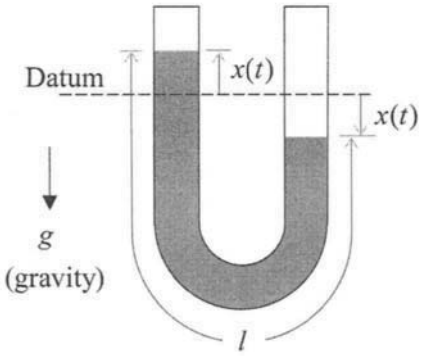
$$\ddot{\theta}(t) + \frac{g}{l}\theta(t) = 0; \quad \omega_n = \sqrt{\frac{g}{l}}$$

Disk-shaft system



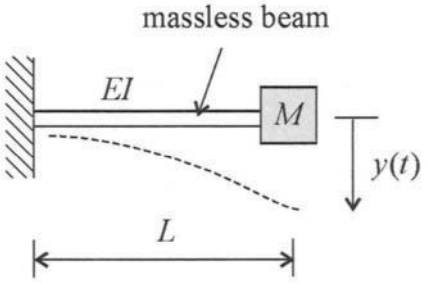
$$I_D\ddot{\theta}(t) + \frac{GJ}{l}\theta(t) = 0; \quad \omega_n = \sqrt{\frac{GJ}{I_D l}}$$

U-tube manometer



$$\ddot{x}(t) + \frac{2g}{l}x(t) = 0; \quad \omega_n = \sqrt{\frac{2g}{l}}$$

Beam with tip mass



$$M\ddot{y}(t) + \frac{3EI}{L^3}y(t) = 0; \quad \omega_n = \sqrt{\frac{3EI}{ML^3}}$$

Eqs. (10.1) and (10.4) are essentially the same. The natural frequency and damping ratio of the system (10.4) are given by

$$\omega_n = \sqrt{\frac{\alpha_K}{\alpha_M}}, \quad \xi = \frac{\alpha_C}{2\sqrt{\alpha_M \alpha_K}} \tag{10.5}$$

Damping Status

(i) Undamped system:	$\xi = 0$	or	$c = 0$
(ii) Underdamped system:	$0 < \xi < 1$	or	$0 < c < 2\sqrt{mk}$
(iii) Critically-damped system:	$\xi = 1$	or	$c = 2\sqrt{mk}$
(iv) Overdamped system:	$\xi > 1$	or	$c > 2\sqrt{mk}$

Characteristic Equation and Roots Consider the homogeneous equation

$$\ddot{x}(t) + 2\xi\omega_n\dot{x}(t) + \omega_n^2 x(t) = 0 \tag{10.6}$$

which is obtained by letting $F(t) = 0$ in Eq. (10.2). Substitute the assumed solution

$$x(t) = Ae^{rt} \tag{10.7}$$

into Eq. (10.6) yields the characteristic equation of the vibrating system

$$r^2 + 2\xi\omega_n r + \omega_n^2 = 0 \tag{10.8}$$

Equation (10.8) has two roots

$$r_{1,2} = -\xi\omega_n \pm \sqrt{\xi^2 - 1}\,\omega_n \tag{10.9}$$

which may be real or complex, depending on the damping in the vibrating system.

(i) Undamped systems ($\xi = 0$): imaginary roots

$$r_{1,2} = \pm j\,\omega_n, \quad j = \sqrt{-1} \tag{10.10a}$$

(ii) Underdamped systems ($0 < \xi < 1$): complex conjugate roots

$$r_{1,2} = -\sigma \pm j\omega_d \tag{10.10b}$$

(iii) Critically-damped systems ($\xi = 1$): repeated negative roots

$$r_{1,2} = -\omega_n \tag{10.10c}$$

(iv) Overdamped system ($\xi > 1$): distinct negative roots

$$r_{1,2} = -\sigma \pm \beta \tag{10.10d}$$

In the previous equations

$$\sigma = \xi\omega_n, \quad \omega_d = \sqrt{1 - \xi^2}\,\omega_n, \quad \beta = \sqrt{\xi^2 - 1}\,\omega_n \tag{10.11}$$

Parameter ω_d is usually called *damped frequency* (rad/s).

System Setup by the Toolbox The Toolbox of this chapter has functions `set1dof` and `set1dof2` for setting up a given 1-DOF system for vibration analysis; see Window 2.1.

Window 2.1. Functions set1dof and set1dof2

MATLAB Functions: **set1dof, set1dof2**

Purpose:
To input system parameters and to display natural frequency and damping status.

Synopsis:

```
set1dof(m,c,k)
set1dof2(wn, zeta, m)
```

Description:
`set1dof(m,c,k)` undertakes the following tasks: (i) it inputs system parameters (mass `m`, damping coefficient `c`, and spring coefficient `k`); (ii) it computes natural frequency and damping ratio; (iii) it computes characteristic roots; and (iv) it shows damping status.

`set1dof2(wn, zeta, m)` has the same functionality as `set1dof` except that it inputs a different set of system parameters: natural frequency `wn`, damping ratio `zeta`, and mass `m`. When `set1dof2(wn, zeta)` is used, the mass is automatically set to `m` = 1. When `set1dof2(wn)` is used, the mass and damping ratio are automatically set to `m` = 1, and `zeta` = 0.

Notes:

Either `set1dof` or `set1dof2` must be executed *before* any other function of vibration analysis can be used. For a vibrating system under different loading cases, however, `set1dof` or `set1dof2` only needs to be called once.

To get the information about the system that has been set up by `set1dof` or `set1dof2`, the MATLAB function **systinfo** can be called at any time.

EXAMPLE 2.1

Consider a vibrating system described by

$$\ddot{x}(t) + 5\dot{x}(t) + 4x(t) = F(t).$$

In the MATLAB command window, typing

```
>> set1dof(1,5,4)
```

yields

```
Vibration Analysis of A One-Degree-of-Freedom System
====================================================
Equation of Motion:
 M*(d/dt)^2*x(t) + C*(d/dt)*x(t) + K*x(t) = F(t)
```

```
M (Mass) = 1
C (Damping Coefficient) = 5
K (Spring Coefficient) = 4

Circular natural frequency, wn = 2 (rad/s)
Natural frequency in Hertz, fn = 0.31831 (Hz)
Natural period Tn = 1/fn = 3.1416 (sec)

Characteristic roots: -4, -1

This is an overdamped system (damping ratio > 1)
Damping ratio, C/(2*M*wn) = 1.25
```

EXAMPLE 2.2

For a 1-DOF vibrating system with mass $m = 2$, damping ratio $\xi = 0.3$, and natural frequency $\omega_n = 5$,

```
>> set1dof2(5, 0.3, 2)
```

yields

```
Vibration Analysis of A One-Degree-of-Freedom System
=====================================================
Equation of Motion:
 M*(d/dt)^2*x(t) + C*(d/dt)*x(t) + K*x(t) = F(t)

 M (Mass) = 2
 C (Damping Coefficient) = 6
 K (Spring Coefficient) = 50

Circular natural frequency, wn = 5 (rad/s)
Natural frequency in Hertz, fn = 0.79577 (Hz)
Natural period Tn = 1/fn = 1.2566 (sec)

Characteristic roots: -1.5+4.7697i, -1.5-4.7697i

This is an underdamped system (0 < damping ratio < 1)
Damping ratio, C/(2*M*wn) = 0.3
Damped frequency, wd = 4.7697 (rad/s)
```

10.3 Time Response

10.3.1 Free Response

The free vibration of a 1-DOF system is described by

$$\begin{aligned} &m\ddot{x}(t) + c\dot{x}(t) + kx(t) = 0, \quad t \geq 0 \\ &x(0) = x_0, \quad \dot{x}(0) = v_0 \end{aligned} \tag{10.12}$$

where x_0 and v_0 are initial displacement and velocity, respectively. The solution to Eq. (10.12) is of the form

$$x(t) = a_1 e^{r_1 t} + a_2 e^{r_2 t} \tag{10.13}$$

where r_1 and r_2 are the characteristic roots given in Eq. (10.9), and a_1 and a_2 are coefficients to be determined by the initial conditions. With Eqs. (10.12) and (10.13), the free response subject to initial disturbances (x_0, v_0) is obtained in the following damping cases:

Case 1. Undamped systems ($\xi = 0$)

$$\begin{aligned} x(t) &= x_0 \cos \omega_n t + \frac{v_0}{\omega_n} \sin \omega_n t \\ \dot{x}(t) &= v_0 \cos \omega_n t - \omega_n x_0 \sin \omega_n t \end{aligned} \tag{10.14}$$

Case 2. Underdamped systems ($0 < \xi < 1$)

$$\begin{aligned} x(t) &= e^{-\sigma t} \left(x_0 \cos \omega_d t + \frac{v_0 + \sigma x_0}{\omega_d} \sin \omega_d t \right) \\ \dot{x}(t) &= e^{-\sigma t} \left(v_0 \cos \omega_d t - \frac{\omega_n^2 x_0 + \sigma v_0}{\omega_d} \sin \omega_d t \right) \end{aligned} \tag{10.15}$$

where $\omega_d = \sqrt{1 - \xi^2}\, \omega_n$ and $\sigma = \xi\, \omega_n$.

Case 3. Critically-damped systems ($\xi = 1$)

$$\begin{aligned} x(t) &= e^{-\omega_n t} \left(x_0 + (v_0 + \omega_n x_0)\, t \right) \\ \dot{x}(t) &= e^{-\omega_n t} \left(v_0 - \omega_n \left(v_0 + \omega_n x_0 \right) t \right) \end{aligned} \tag{10.16}$$

Case 4. Overdamped systems ($\xi > 1$)

$$\begin{aligned} x(t) &= e^{-\sigma t} \left(x_0 \cosh \beta t + \frac{v_0 + \sigma x_0}{\beta} \sinh \beta t \right) \\ \dot{x}(t) &= e^{-\sigma t} \left(v_0 \cosh \beta t - \frac{\omega_n^2 x_0 + \sigma v_0}{\beta} \sinh \beta t \right) \end{aligned} \tag{10.17}$$

where $\beta = \sqrt{\xi^2 - 1}\, \omega_n$ and $\sigma = \xi \omega_n$.

Solution by the Toolbox The Toolbox of this chapter has a function `freevib` for computing free vibration; see Window 3.1.

Window 3.1. Function freevib

MATLAB Function: **freevib**

Purpose:
To plot free response of a 1-DOF system.

Synopsis:

```
freevib(x0,v0)
freevib
freevib(x0,v0, tf, npts)
[y, t] = freevib(x0,v0, tf, npts)
```

Description:
`freevib(x0,v0)` plots free response subject to initial displacement `x0` and initial velocity `v0`.

`freevib` plots free response for `x0` = 1 and `v0` = 0.

`freevib(x0,v0, tf, npts)` plots free response with the specified total simulation time `tf`, and the number `npts` of time points in the region $0 \leq t \leq$ `tf`. By default `tf` = $8\pi/\omega_n$ and `npts` = 101. Also, `freevib(x0,v0, [ ], npts)` or `freevib(x0,v0, tf)` can be used, where `tf` is replaced by [ ] as a placeholder.

`[y, t] = freevib(x0,v0, tf, npts)` returns computed free response in a two-row matrix `y`, and the times of computation in a vector `t`. Of the matrix `y`, the first row `y(1, :)` contains displacement, and the second row `y(2, :)` contains velocity. With `y` and `t`, the time history of the vibrating system can be plotted by the MATLAB function `plot` as follows:

```
plot(t, y(1,:))   for displacement
plot(t, y(2,:))   for velocity
```

Also, function `plotvib` from this Toolbox (Window 3.12 in Section 10.3.8) can be used to plot free response.

EXAMPLE 3.1

The response of the system

$$2\ddot{x}(t) + 3\dot{x}(t) + 10x(t) = 0$$
$$x(0) = 0.1, \quad \dot{x}(0) = -0.5$$

is obtained by

```
>> set1dof(2,3,10)
>> freevib(0.1, -0.5)
```

which yields the plot of displacement versus time in Fig. 10.3.1.

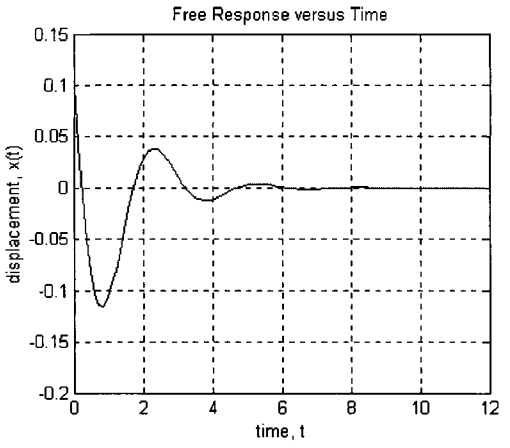


FIGURE 10.3.1

EXAMPLE 3.2

For the undamped system

$$\ddot{x}(t) + 4x(t) = 0$$
$$x(0) = 1, \quad \dot{x}(0) = 0$$

```
>> set1dof2(2)
>> freevib
```

plots its free response in Fig. 10.3.2.

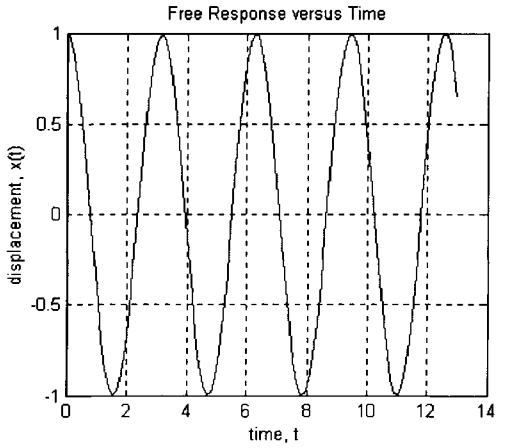


FIGURE 10.3.2

10.3.2 Forced Response

The forced response of a 1-DOF system is described by

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t), \quad t \geq 0$$
$$x(0) = 0, \quad \dot{x}(0) = 0 \tag{10.18}$$

where $F(t)$ is an external force. Here, zero initial conditions have been assumed for two reasons. First, the free response and forced response would be totally decoupled this way. In other words, the total response of a system subject to both external and initial disturbances can be obtained by simple summation of the forced response and the free response. Second, the forced response so determined is useful in analysis of a system under multiple loads by the principle of superposition, as discussed in Section 10.3.6.

The Toolbox of this chapter provides a function `forcedvib` for computing the forced response of a 1-DOF system; see Window 3.2. Refer to Section 10.4.1 for solution methods for this vibration problem.

Window 3.2. Function `forcedvib`

MATLAB Function: **`forcedvib`**

Purpose:
To plot forced response of a 1-DOF system.

Synopsis:

```
forcedvib
forcedvib(Load_Spec)
forcedvib(Load_Spec, tf, npts)
[y, t] = forcedvib(Load_Spec, tf, npts)
```

Description:
`forcedvib` plots the forced response of a system subject to a unit step load, $F(t) = 1$.

`forcedvib(Load_Spec)` plots the forced response subject to an external load that is specified by a vector `Load_Spec`, according to Table 10.3.1.

`forcedvib(Load_Spec, tf, npts)` plots forced response with the specified total simulation time `tf`, and the number `npts` of time points in the region $0 \leq t \leq$`tf`. By default, `tf` $= 8\pi/\omega_n$, and `npts` $= 101$. Also, `forcedvib(Load_Spec, tf)` or `forcedvib(Load_Spec, [ ], npts)` is valid, where `tf` can be replaced by `[ ]` as a placeholder.

Window 3.2. Function forcedvib (*Continued*)

`[y, t] = forcedvib(Load_Spec, tf, npts)` returns the computed forced response in a two-row matrix `y`, and the times of simulation in a vector `t`. Of the matrix `y`, the first row `y(1, :)` contains displacement, and the second row `y(2, :)` contains velocity. With `y` and `t`, the time history of the vibrating system can be plotted by the MATLAB function `plot` as follows:

```
plot(t, y(1,:))   for displacement
plot(t, y(2,:))   for velocity
```

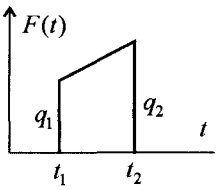
Also, function `plotvib` from this Toolbox (Window 3.12 in Sec. 10.3.8) can be used to plot forced response.

The argument `Load_Spec` of function `forcedvib` is a vector specifying an external load. Listed in Table 10.3.1 are six types of external loads considered in the Toolbox. In general, vector `Load_Spec` is of the form

```
Load_Spec = [load_ID par1 par2 ...]
```

where the first element is the load type identification number (`load_ID` $= 0, 1, \ldots, 5$), and the rest are the parameters that describe the magnitude and pattern of the forcing function. For the mathematical expressions of the forced response subject to the loads listed in Table 10.3.1 and beyond, refer to Section 10.3.3.

TABLE 10.3.1 External Loads and Load Specification Vector `Load_Spec`

External Load	*F(t)*	`Load_Spec`
(L0) Impulse	$I_0\delta(t)$	[0 I_0]
(L1) Step	F_0	[1 F_0]
(L2) Ramp	$a_0 t$	[2 a_0]
(L3) Exponential	$A e^{bt}$	[3 A b]
(L4) Sinusoidal	$A\sin(\omega t + \phi_0)$	[4 A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
(L5) Pulse excitation		[5 t_1 q_1 t_2 q_2]

EXAMPLE 3.3

An undamped system with parameters $m = 1$ and $k = 4$ is subject to $F(t) = 2\sin(2t)$. Because the excitation frequency coincides with the natural frequency of the system, i.e., $\omega = \omega_n = 2$, resonance occurs. Executing the commands

```
>> set1dof(1, 0, 4)
>> forcedvib([4 2 2 0],30)
```

shows this phenomenon in Fig. 10.3.3. When the excitation frequency is close to the natural frequency, say $F(t) = 2\sin(1.8t)$,

```
>> forcedvib([4 2 1.8 0],80,801)
```

exhibits the beating phenomenon in Fig. 10.3.4. (See Section 10.3.3 for an explanation of beating.)

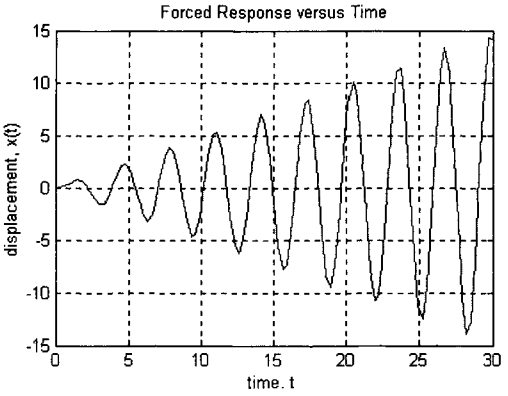


FIGURE 10.3.3 Resonant vibration.

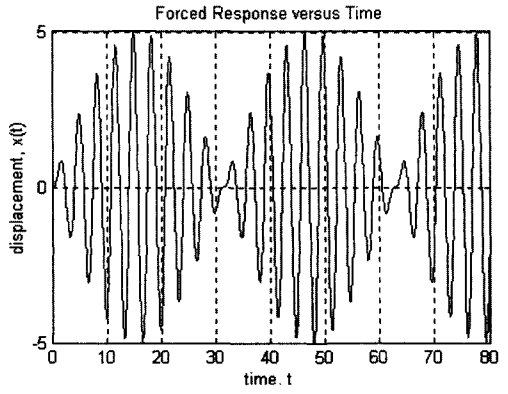


FIGURE 10.3.4 Beating phenomenon.

EXAMPLE 3.4

An underdamped system of parameters $m = 3$, $\omega_n = 18$, and $\xi = 0.24$ is subject to an impulse $F(t) = 15\delta(t)$, where $\delta(t)$ is the delta function. By the following MATLAB commands:

```
>> setldof2(18, 0.24, 3)
>> forcedvib([0 15])
```

the displacement of the system is plotted in Fig. 10.3.5.

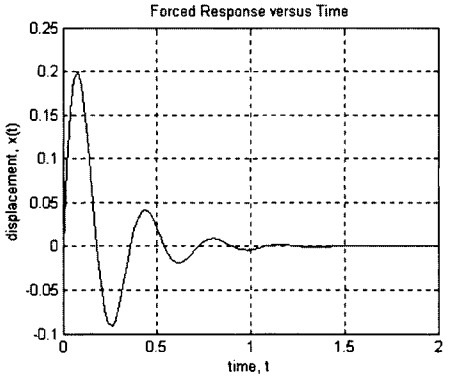


FIGURE 10.3.5

EXAMPLE 3.5

A system with parameters $m = 1$, $c = 1$, and $k = 4$ is subject to a pulse excitation shown in Fig. 10.3.6. The commands

```
>> setldof(1, 1, 4)
>> load = [5 2 -2 6 3];
>> forcedvib(load, 20, 401)
```

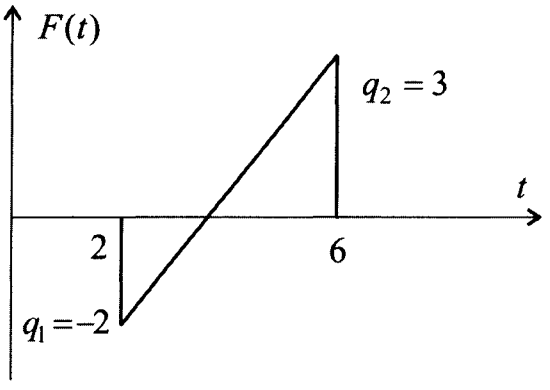


FIGURE 10.3.6

yield the forced response in Fig. 10.3.7.

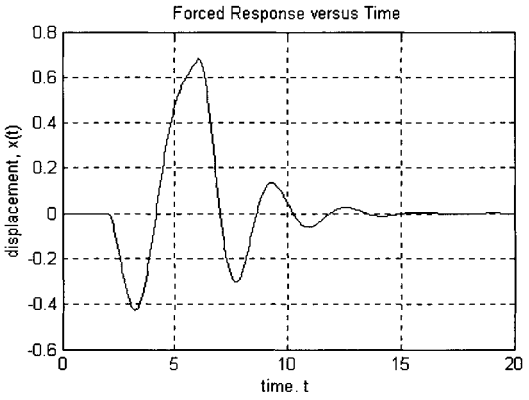


FIGURE 10.3.7

10.3.3 Specific Forcing Functions

In this section, the forced response of a 1-DOF vibrating system subject to the forcing functions listed in Table 10.3.1 is presented. The results presented herein can be obtained by the analytical solution methods given in Section 10.4.1. In sequel, the following parameters will be used

$$\sigma = \xi\omega_n, \quad \omega_d = \sqrt{1-\xi^2}\,\omega_n, \quad \beta = \sqrt{\xi^2-1}\,\omega_n. \tag{10.19}$$

Response to Impulse Force Under the impulse load

$$F(t) = I_0\delta(t) \tag{10.20}$$

where $\delta(t)$ is the delta function, the forced response is obtained in three damping cases:

Case 1. Undamped systems ($\xi = 0$)

$$\begin{aligned} x(t) &= \frac{I_0}{m\omega_n}\sin\omega_n t \\ \dot{x}(t) &= \frac{I_0}{m}\cos\omega_n t \end{aligned} \tag{10.21a}$$

Case 2. Underdamped systems ($0 < \xi < 1$)

$$\begin{aligned} x(t) &= \frac{I_0}{m\omega_d}e^{-\sigma t}\sin\omega_d t \\ \dot{x}(t) &= \frac{I_0}{m}e^{-\sigma t}\left(\cos\omega_d t - \frac{\sigma}{\omega_d}\sin\omega_d t\right) \end{aligned} \tag{10.21b}$$

Case 3. Critically-damped systems ($\xi = 1$)

$$x(t) = \frac{I_0}{m} t\, e^{-\omega_n t}$$
$$\dot{x}(t) = \frac{I_0}{m} e^{-\omega_n t} (1 - \omega_n t) \tag{10.21c}$$

Case 4. Overdamped systems ($\xi > 1$)

$$x(t) = \frac{I_0}{m\beta} e^{-\sigma t} \sinh \beta t$$
$$\dot{x}(t) = \frac{I_0}{m} e^{-\sigma t} \left(\cosh \beta t - \frac{\sigma}{\beta} \sinh \beta t \right) \tag{10.21d}$$

Computation of impulse response by the Toolbox is summarized in Window 3.3.

Window 3.3. How to Plot Impulse Response

External Load: Impulse $F(t) = I_0 \delta(t)$

MATLAB Command:

```
forcedvib([0 I0])
```

Response to Step Force Under the step (constant) load

$$F(t) = F_0 \tag{10.22}$$

the forced vibration is given in the following damping cases.

Case 1. Undamped systems ($\xi = 0$)

$$x(t) = \frac{F_0}{k} (1 - \cos \omega_n t)$$
$$\dot{x}(t) = \frac{F_0}{m\omega_n} \sin \omega_n t \tag{10.23a}$$

Case 2. Underdamped systems ($0 < \xi < 1$)

$$x(t) = \frac{F_0}{k} \left\{ 1 - e^{-\sigma t} \left(\cos \omega_d t + \frac{\sigma}{\omega_d} \sin \omega_d t \right) \right\}$$
$$\dot{x}(t) = \frac{F_0}{m\omega_d} e^{-\sigma t} \sin \omega_d t \tag{10.23b}$$

Case 3. Critically-damped systems ($\xi = 1$)

$$x(t) = \frac{F_0}{k}\left\{1 - e^{-\omega_n t}(1 + \omega_n t)\right\}$$
$$\dot{x}(t) = \frac{F_0}{m} t\, e^{-\omega_n t} \tag{10.23c}$$

Case 4. Overdamped systems ($\xi > 1$)

$$x(t) = \frac{F_0}{k}\left\{1 - e^{-\sigma t}\left(\cosh \beta t + \frac{\sigma}{\beta}\sinh \beta t\right)\right\}$$
$$\dot{x}(t) = \frac{F_0}{m\beta} e^{-\sigma t} \sinh \beta t \tag{10.23d}$$

In dynamic analysis and control of mechanical systems, the step response of an underdamped system ($0 < \xi < 1$) is often characterized by maximum overshoot M_p, rise time t_r, and settling time t_s (Reference 7); see Fig. 10.3.8. These parameters are defined as follows.

Maximum overshoot

$$M_p = \frac{x(t_p) - x_{ss}}{x_{ss}} = e^{-\pi\xi/\sqrt{1-\xi^2}} \tag{10.24}$$

where $t_p = \pi/\omega_d$ is the peak time at which the step response reaches its maximum, and $x_{ss} = F_0/k$ is the final value or steady-state of the response.

Rise time

$$t_r = \frac{1}{\omega_d}\left(\pi - \tan^{-1}(\omega_d/\sigma)\right) \tag{10.25}$$

which is the time required for the system response to rise from 0% to 100% of its steady-state value $x_{ss} = F_0/k$.

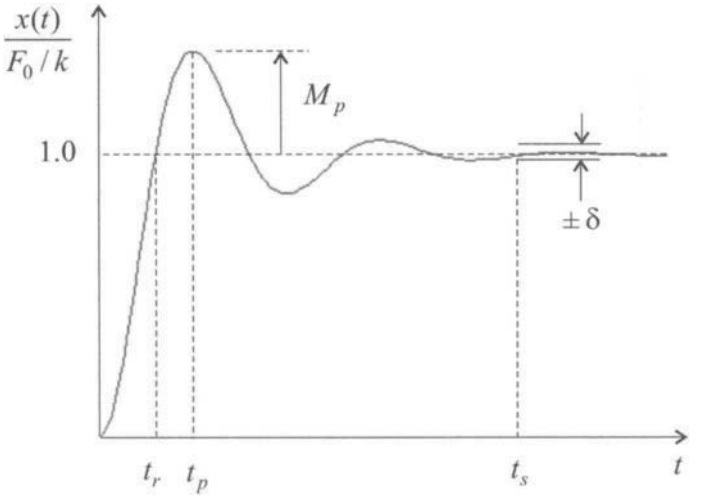


FIGURE 10.3.8 Specification of a step response of an underdamped system.

Settling time

$$t_s = -\frac{1}{\xi\omega_n}\ln\left(\delta\sqrt{1-\xi^2}\right) \tag{10.26}$$

which is the time required for the system response to settle within a certain percentage ($\pm\delta\times 100\%$) of its final value F_0/k. Here, δ defines a band of allowable tolerance. In practice, settling time is approximated as follows

$$\begin{aligned} &5\%\text{ criterion } (\delta = 0.05): \quad t_s = \frac{3}{\xi\omega_n} \\ &2\%\text{ criterion } (\delta = 0.02): \quad t_s = \frac{4}{\xi\omega_n} \\ &1\%\text{ criterion } (\delta = 0.01): \quad t_s = \frac{4.6}{\xi\omega_n} \end{aligned} \tag{10.27}$$

The parameters M_p and t_s are related to the system parameters by

$$\frac{(\ln M_p)^2}{\pi^2 + (\ln M_p)^2} = \xi^2 = \frac{c^2}{4mk}, \quad t_s = \frac{\chi}{\xi\omega_n} = \frac{\chi}{(c/2m)} \tag{10.28}$$

where $\chi = 3$, 4, and 4.6 for the 5%, 2%, and 1% criteria, respectively.

Computation of step response by the Toolbox is given in Window 3.4.

Window 3.4. How to Plot Step Response

External Load: Step load $F(t) = F_0$

MATLAB Command:

```
forcedvib([1 F0])
```

When a step load is applied to an underdamped system ($0 < \xi < 1$), function `forcedvib` automatically displays the information of maximum overshoot, rise time, and settling time. The information of those parameters is also retrievable by typing `systinfo` in the MATLAB command window. In addition, for given natural frequency `wn` and damping ratio `zeta`, the maximum overshoot, rise time, and settling time of step response can be computed by

```
>> stepcharact(wn, zeta)
```

where `stepcharact` is a function from the Toolbox.

Response to Ramp Force Under the ramp load

$$F(t) = a_0 t \tag{10.29}$$

the system response is given in the following damping cases:

Case 1. Undamped systems ($\xi = 0$)

$$x(t) = \frac{a_0}{k\omega_n}(\omega_n t - \sin\omega_n t)$$
$$\dot{x}(t) = \frac{a_0}{k}(1 - \cos\omega_n t) \tag{10.30a}$$

Case 2. Underdamped systems ($0 < \xi < 1$)

$$x(t) = \frac{a_0}{k}\left\{t - \frac{c}{k} + e^{-\sigma t}\left(\frac{c}{k}\cos\omega_d t + \frac{1}{\omega_d}(\sigma\frac{c}{k} - 1)\sin\omega_d t\right)\right\}$$
$$\dot{x}(t) = \frac{a_0}{k}\left\{1 - e^{-\sigma t}\left(\cos\omega_d t + \frac{\sigma}{\omega_d}\sin\omega_d t\right)\right\} \tag{10.30b}$$

Case 3. Critically-damped systems ($\xi = 1$)

$$x(t) = \frac{a_0}{k}\left\{t - \frac{c}{k} + e^{-\omega_n t}\left(\frac{c}{k} + (\omega_n\frac{c}{k} - 1)t\right)\right\}$$
$$\dot{x}(t) = \frac{a_0}{k}\left\{1 - e^{-\omega_n t}(1 + \omega_n t)\right\} \tag{10.30c}$$

Case 4. Overdamped systems ($\xi > 1$)

$$x(t) = \frac{a_0}{k}\left\{t - \frac{c}{k} + e^{-\sigma t}\left(\frac{c}{k}\cosh\beta t + \frac{1}{\beta}(\sigma\frac{c}{k} - 1)\sinh\beta t\right)\right\}$$
$$\dot{x}(t) = \frac{a_0}{k}\left\{1 - e^{-\sigma t}\left(\cosh\beta t + \frac{\sigma}{\beta}\sinh\beta t\right)\right\} \tag{10.30d}$$

Computation of ramp response by the Toolbox is summarized in Window 3.5.

Window 3.5. How to Plot Ramp Response

External Load: Ramp load $F(t) = a_0 t$

MATLAB Command:

```
forcedvib([2 a0])
```

Response to Exponential Force Under the load of the exponential form

$$F(t) = Ae^{bt} \tag{10.31}$$

the forced response depends on the value of $b^2m + bc + k$. If $b^2m + bc + k \neq 0$, the forced response is as follows:

Case 1. Undamped systems ($\xi = 0$)

$$\begin{aligned} x(t) &= \frac{A}{b^2m + k}\left(e^{bt} - \cos\omega_n t - \frac{b}{\omega_n}\sin\omega_n t\right) \\ \dot{x}(t) &= \frac{A}{b^2m + k}\left(be^{bt} - b\cos\omega_n t + \omega_n \sin\omega_n t\right) \end{aligned} \tag{10.32a}$$

Case 2. Underdamped systems ($0 < \xi < 1$)

$$\begin{aligned} x(t) &= \frac{A}{b^2m + bc + k}\left\{e^{bt} - e^{-\sigma t}\left(\cos\omega_d t + \frac{b+\sigma}{\omega_d}\sin\omega_d t\right)\right\} \\ \dot{x}(t) &= \frac{A}{b^2m + bc + k}\left\{be^{bt} - be^{-\sigma t}\cos\omega_d t + \frac{\omega_n^2 + \sigma b}{\omega_d}e^{-\sigma t}\sin\omega_d t\right\} \end{aligned} \tag{10.32b}$$

Case 3. Critically-damped systems ($\xi = 1$)

$$\begin{aligned} x(t) &= \frac{A}{b^2m + bc + k}\left\{e^{bt} - e^{-\omega_n t}\left[1 + (b + \omega_n)\cdot t\right]\right\} \\ \dot{x}(t) &= \frac{A}{b^2m + bc + k}\left\{be^{bt} - be^{-\omega_n t} + (b + \omega_n)\,\omega_n t e^{-\omega_n t}\right\} \end{aligned} \tag{10.32c}$$

Case 4. Overdamped systems ($\xi > 1$)

$$\begin{aligned} x(t) &= \frac{A}{b^2m + bc + k}\left\{e^{bt} - e^{-\sigma t}\left(\cosh\beta t + \frac{b+\sigma}{\beta}\sinh\beta t\right)\right\} \\ \dot{x}(t) &= \frac{A}{b^2m + bc + k}\left\{be^{bt} - be^{-\sigma t}\cosh\beta t + \frac{\omega_n^2 + \sigma b}{\beta}e^{-\sigma t}\sinh\beta t\right\} \end{aligned} \tag{10.32d}$$

When b is identical to one of the characteristic roots given in Eq. (10.9), $b^2m + bc + k = 0$. The forced response in this situation is as follows.

Case 1. $b = -\sigma = -\xi\omega_n$ (for critically damped systems):

$$\begin{aligned} x(t) &= \frac{A}{2}t^2 e^{-\sigma t} \\ \dot{x}(t) &= \tfrac{A}{2}(2t - \sigma t^2)e^{-\sigma t} \end{aligned} \tag{10.33a}$$

Case 2. $b = -\xi\omega_n + \sqrt{\xi^2 - 1}\,\omega_n$ or $-\xi\omega_n - \sqrt{\xi^2 - 1}\,\omega_n$ (for overdamped systems):

$$\begin{aligned} x(t) &= \frac{A}{c + 2bm}\left\{te^{bt} - e^{-\sigma t}\frac{1}{\beta}\sinh\beta t\right\} \\ \dot{x}(t) &= \frac{A}{c + 2bm}\left\{(1 + bt)e^{bt} - e^{-\sigma t}\left(\cosh\beta t - \frac{\sigma}{\beta}\sinh\beta t\right)\right\} \end{aligned} \tag{10.33b}$$

Because b is real, $b^2m + bc + k \neq 0$ for undamped and underdamped systems.

Solution of forced response to an exponential load is given in Window 3.6.

Window 3.6. How to Plot Response to Exponential Load

External Load: Ramp load $F(t) = Ae^{bt}$

MATLAB Command:

```
forcedvib([3 A b])
```

Response to Sinusoidal Excitation Under the sinusoidal load

$$F(t) = A\sin(\omega t + \phi_0) \tag{10.34}$$

where ω is the excitation frequency, the solution of Eq. (10.18) exhibits either nonresonant vibration or resonant vibration.

Nonresonant Vibration ($\xi > 0$ or $\omega \neq \omega_n$)

Case 1. Undamped systems ($\xi = 0$)

$$\begin{aligned} x(t) &= A \cdot h \cdot \left\{ \sin(\omega t + \phi_0 - \psi) - \sin(\phi_0 - \psi)\cos\omega_n t - \frac{\omega}{\omega_n}\cos(\phi_0 - \psi)\sin\omega_n t \right\} \\ \dot{x}(t) &= A \cdot h \cdot \{\omega\cos(\omega t + \phi_0 - \psi) - \omega\cos(\phi_0 - \psi)\cos\omega_n t + \omega_n \sin(\phi_0 - \psi)\sin\omega_n t\} \end{aligned} \tag{10.35a}$$

Case 2. Underdamped systems ($0 < \xi < 1$)

$$\begin{aligned} x(t) &= A \cdot h\sin(\omega t + \phi_0 - \psi) - A \cdot he^{-\sigma t}\sin(\phi_0 - \psi)\cos\omega_d t \\ &\quad - A \cdot he^{-\sigma t}\frac{\omega\cos(\phi_0 - \psi) + \sigma\sin(\phi_0 - \psi)}{\omega_d}\sin\omega_d t \\ \dot{x}(t) &= A \cdot h\omega\cos(\omega t + \phi_0 - \psi) - A \cdot h\omega e^{-\sigma t}\cos(\phi_0 - \psi)\cos\omega_d t \\ &\quad + A \cdot h\, e^{-\sigma t}\frac{\sigma\omega\cos(\phi_0 - \psi) + \omega_n^2\sin(\phi_0 - \psi)}{\omega_d}\sin\omega_d t \end{aligned} \tag{10.35b}$$

Case 3. Critically-damped systems

$$\begin{aligned} x(t) &= A \cdot h\sin(\omega t + \phi_0 - \psi) - A \cdot h\, e^{-\omega_n t}\sin(\phi_0 - \psi) \\ &\quad - A \cdot h\, e^{-\omega_n t}\,[\omega\cos(\phi_0 - \psi) + \omega_n\sin(\phi_0 - \psi)] \cdot t \\ \dot{x}(t) &= A \cdot h\,\omega\cos(\omega t + \phi_0 - \psi) - A \cdot h\,\omega\, e^{-\omega_n t}\cos(\phi_0 - \psi) \\ &\quad + A \cdot h\, e^{-\omega_n t}\,[\omega\cos(\phi_0 - \psi) + \omega_n\sin(\phi_0 - \psi)] \cdot \omega_n t \end{aligned} \tag{10.35c}$$

Case 4. Overdamped systems

$$
\begin{aligned}
x(t) &= A \cdot h \cdot \sin(\omega t + \phi_0 - \psi) - A \cdot h \cdot e^{-\sigma t} \sin(\phi_0 - \psi) \cosh \beta t \\
&\quad - A \cdot h \cdot e^{-\sigma t} \frac{\omega \cos(\phi_0 - \psi) + \sigma \sin(\phi_0 - \psi)}{\beta} \sinh \beta t \\
\dot{x}(t) &= A \cdot h\omega \cos(\omega t + \phi_0 - \psi) - A \cdot h\omega e^{-\sigma t} \cos(\phi_0 - \psi) \cosh \beta t \\
&\quad + A \cdot h \cdot e^{-\sigma t} \frac{\sigma\omega \cos(\phi_0 - \psi) + \omega_n^2 \sin(\phi_0 - \psi)}{\beta} \sinh \beta t
\end{aligned}
\tag{10.35d}
$$

In the previous equations

$$
h = \frac{1}{\sqrt{(k - \omega^2 m)^2 + c^2\omega^2}}, \quad \psi = \tan^{-1}\left(\frac{c\omega}{k - m\omega^2}\right) \tag{10.36}
$$

Resonant Vibration ($\xi = 0$ and $\omega = \omega_n$)

Resonance occurs when the system is undamped and the excitation frequency coincides with the system natural frequency. In this case, the forced response is

$$
\begin{aligned}
x(t) &= \frac{A}{2\omega_n^2 m} [\cos(\phi_0) \sin \omega_n t - \omega_n t \cos(\omega_n t + \phi_0)] \\
\dot{x}(t) &= \frac{A}{2\omega_n m} [\sin(\phi_0) \sin \omega_n t + \omega_n t \cdot \sin(\omega_n t + \phi_0)]
\end{aligned}
\tag{10.37}
$$

Beating Phenomenon

For an undamped system, if the excitation frequency is close to, but not equal to, the natural frequency, a phenomenon known as *beating* takes place. This phenomenon, which has been shown in Example 3.3, is explained mathematically as follows. Let the external load be

$$
F(t) = F_0 \cos \omega t, \quad \text{with } \omega = \omega_n - \varepsilon
$$

where ε is a small positive number. As can be seen, the excitation frequency is slightly less than the natural frequency. By Eq. (10.35a), the displacement of the system is

$$
\begin{aligned}
x(t) &= \frac{F_0}{m\varepsilon(2\omega_n - \varepsilon)} \{\cos(\omega_n - \varepsilon)t - \cos \omega_n t\} \\
&= \frac{2F_0}{m\varepsilon(2\omega_n - \varepsilon)} \sin\left(\frac{2\omega_n - \varepsilon}{2} t\right) \cdot \sin \frac{\varepsilon}{2} t
\end{aligned}
$$

Because ε is much smaller than ω_n,

$$
\begin{aligned}
m\varepsilon(2\omega_n - \varepsilon) &\approx 2m\omega_n\varepsilon \\
\sin\left(\frac{2\omega_n - \varepsilon}{2} t\right) &\approx \sin \omega t
\end{aligned}
$$

This leads to

$$
x(t) = A_m(t) \sin \omega t, \quad \text{with } A_m(t) = \frac{F_0}{m\omega_n\varepsilon} \sin \frac{\varepsilon}{2} t
$$

Thus, the motion of the system can be viewed as a sine wave with the amplitude $A_m(t)$ varying slowly in a sinusoidal fashion. The time between two adjacent points of maximum or zero amplitude is called the *period of beating*, and is given by

$$T_b = \frac{2\pi}{\varepsilon} = \frac{2\pi}{\omega_n - \omega}.$$

The frequency of beating is then defined as

$$\omega_b = \frac{2\pi}{T_b} = \varepsilon = \omega_n - \omega.$$

Additionally, as $\varepsilon \to 0$, the system response is $x(t) = \frac{F_0 t}{2m\omega_n} \sin \omega_n t$, which is the resonant response given by Eq. (10.37).

Solution of the forced response by the Toolbox is summarized in Window 3.7.

Window 3.7. How to Plot Response to Sinusoidal Excitation

External Load: Sinusoidal load $F(t) = A \sin(\omega t + \phi_0)$

MATLAB Command:

```
forcedvib([4 A  ω  φ0])
```

Response to Pulse Excitation Shown in Fig. 10.3.9 is a pulse forcing function, which mathematically is expressed by

$$F(t) = \begin{cases} q_1 + \dfrac{q_2 - q_1}{t_2 - t_1}(t - t_1), & t_1 \le t \le t_2 \\ 0, & \text{otherwise} \end{cases} \tag{10.38a}$$

The forced response can be obtained by superposition. To this end, the pulse is decomposed into several delayed step and ramp functions

$$\begin{aligned} F(t) &= F_1(t) + F_2(t) + F_3 + F_4(t) \\ &= q_1 u(t - t_1) + k(t - t_1)u(t - t_1) - k(t - t_2)u(t - t_2) - q_2 u(t - t_2) \end{aligned} \tag{10.38b}$$

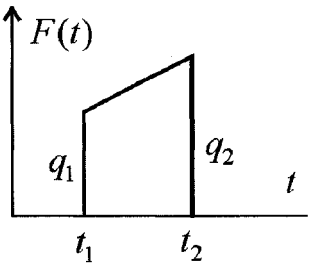


FIGURE 10.3.9 A pulse forcing function.

where $k = (q_2 - q_1)/(t_2 - t_1)$, and $u(t - T)$ is a delayed step function defined as follows

$$u(t-T) = \begin{cases} 1 & \text{for } t \geq T \\ 0 & \text{for } t < T \end{cases} \tag{10.39}$$

By the principle of superposition, the forced response is given by

$$x(t) = x_1(t) + x_2(t) + x_3(t) + x_4(t) \tag{10.40}$$

where $x_j(t)$ is the response due to the forcing function $F_j(t)$. Forced response of a vibrating system under delayed forcing functions is discussed in Sec. 10.3.5.

The forced response of a vibrating system to the pulse can be computed by the Toolbox according to Window 3.8; see Example 3.5.

Window 3.8. How to Plot Response to Pulse Excitation

External Load: Pulse excitation, Eq. (10.38a)

MATLAB Command:

```
forcedvib([5 t1 q1 t2 q2])
```

10.3.4 Total Response

Now consider a system subject to both external and initial disturbances:

$$\begin{gathered} m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t), \quad t \geq 0 \\ x(0) = x_0, \quad \dot{x}(0) = v_0 \end{gathered} \tag{10.41}$$

The solution to Eq. (10.41), which is the total response of the system, can be written as

$$x(t) = x_{Free}(t) + x_{Forced}(t) \tag{10.42}$$

where $x_{Free}(t)$ is the free vibration governed by Eq. (10.12), and $x_{Forced}(t)$ is the forced vibration described by Eq. (10.18).

The free and forced response can be individually obtained by functions `freevib` and `forcedvib`. The Toolbox of this chapter provides function `compltvib` for computing the total response of a vibrating system subject to both initial and external disturbances; see Window 3.9.

Window 3.9. Function `compltvib`

MATLAB Function: **`compltvib`**

Purpose:
To plot total response due to both initial and external disturbances.

Window 3.9. Function `compltvib` (*Continued*)

Synopsis:

```
compltvib(x0, v0, Load_Spec)
[y, tt] = compltvib(x0, v0, Load_Spec, tf, npts)
```

Description:

`compltvib(x0, v0, Load_Spec)` plots the total response of a vibrating system due to an external load specified by vector `Load_Spec` and initial disturbance `x0` and `v0`. The specification of `Load_Spec` follows that for function `forcedvib` (see Table 10.3.1).

`[y, tt] = compltvib(x0, v0, Load_Spec, tf, npts)` returns the computed response in a two-row matrix `y`, and the times of simulation in the vector `tt`. Of `y`, the first row `y(1, :)` contains displacement, and the second row `y(2, :)` contains velocity. Here, `tf` is the total time of simulation, and `npts` is the number of times in the region $0 \le t \le$ `tf`. By default, `tf` $= 8\pi/\omega_n$ and `npts` $= 101$. `tf` can be replaced by `[ ]` as a placeholder for its default value.

EXAMPLE 3.6

A vibrating system of parameters $m = 8$, $\omega_n = 25$, and $\xi = 0.15$ is subject to an exponential load $F(t) = 120e^{-0.25t}$ and initial disturbances $x(0) = -0.3$, $\dot{x}(0) = 0$. By

```
>> set1dof2(25, 0.15, 8)
>> compltvib(-0.3, 0, [3 120 -0.25])
```

the total response of the system is plotted in Fig. 10.3.10, which can also be obtained by

```
>> y = freevib(-0.3, 0) + forcedvib([3 120 -0.25]);
>> plotvib(y)
```

where function `plotvib` is given in Window 3.12 of Section 10.3.8.

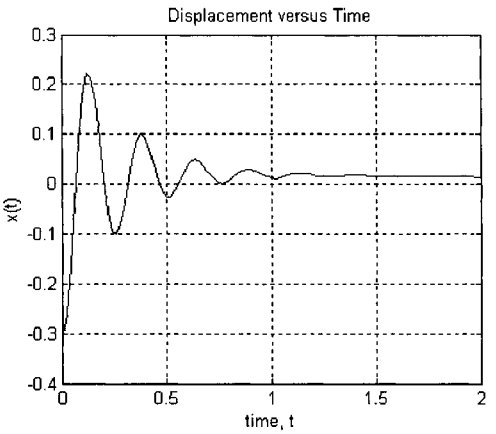


FIGURE 10.3.10

The total response of a system sometimes can also be viewed as a sum of *transient response* and *steady-state response*

$$x(t) = x_{tr}(t) + x_{ss}(t) \tag{10.43}$$

The transient response $x_{tr}(t)$ is excited by initial disturbances and/or sudden application of an external force. In the presence of damping, $x_{tr}(t)$ eventually dies out. The steady-state response $x_{ss}(t)$ is a forced response after a long time has elapsed. As an example, the total response of an underdamped system subject to initial disturbances and a step-forcing function, by Eq. (10.23b), is

$$x(t) = e^{-\sigma t}\left(x_0 \cos \omega_d t + \frac{v_0 + \sigma x_0}{\omega_d} \sin \omega_d t\right) - \frac{F_0}{k} e^{-\sigma t}\left(\cos \omega_d t + \frac{\sigma}{\omega_d} \sin \omega_d t\right) + \frac{F_0}{k}$$

where the first two terms represent the transient response and the last term is the steady-state response.

10.3.5 Delayed Forcing Functions

In some cases, external forces are not applied initially, but some time later. To deal with this situation, a delayed or time-shifted forcing function is introduced:

$$F(t - T_d)\, u(t - T_d) = \begin{cases} F(t - T_d) & \text{for } t \geq T_d \\ 0 & \text{for } t < T_d \end{cases} \tag{10.44}$$

where T_d is a nonnegative time delay parameter, $F(t)$ is the corresponding nondelayed forcing function, and $u(t - T_d)$ is a unite step function as defined in Eq. (10.39). Under such a delayed forcing function, the forced response of a vibrating system can be obtained by shifting (with delay T_d) the response subjected to the nondelayed load $F(t)$. For instance, the response of an undamped system subject to a step force $F(t) = F_0$ is

$$x(t) = \frac{F_0}{k}\,(1 - \cos \omega_n t)\,.$$

For the same system subject to a delayed step function $F_0 u(t - T_d)$, its response is

$$x(t) = \begin{cases} \dfrac{F_0}{k}\,(1 - \cos \omega_n (t - T_d))\,, & \text{for } t \geq T_d \\ 0, & \text{for } t < T_d \end{cases}$$

In vibration analysis by the Toolbox, the forcing functions (L0) to (L4) given in Table 10.3.1 can be easily time-shifted by adding a time delay parameter T_d to the end of the load specification vector; see Table 10.3.2. Solution of the forced response to delayed forcing functions still calls function `forcedvib` or `compltvib`.

TABLE 10.3.2 Time-shifted Forcing Functions and Load Specification Vector Load_Spec

External Load	*Delayed Forcing Function*	Load_Spec
(L0) Impulse	$I_0\delta(t - T_d)$	[0 I_0 T_d]
(L1) Step	$F_0u(t - T_d)$	[1 F_0 T_d]
(L2) Ramp	$a_0(t - T_d)u(t - T_d)$	[2 a_0 T_d]
(L3) Exponential	$Ae^{b(t-T_d)}u(t - T_d)$	[3 A b T_d]
(L4) Sinusoidal	$A\sin(\omega(t - T_d) + \phi_0)\,u(t - T_d)$	[4 A ω ϕ_0 T_d] ω in rad/s, ϕ_0 in degrees

EXAMPLE 3.7

A system with parameters $m = 1$, $\omega_n = 4$, and $\xi = 0.15$ is subject to a delayed impulse $F(t) = 12\delta(t - 10)$ and initial disturbances $x(0) = 2$ and $\dot{x}(0) = 0$. The total response is plotted in Fig. 10.3.11 by

```
>> setldof2(4, 0.15)
>> compltvib(2, 0, [0 12 10], 20, 601);
```

As can be seen, the system response is excited first by the initial displacement, and later on by the impulse.

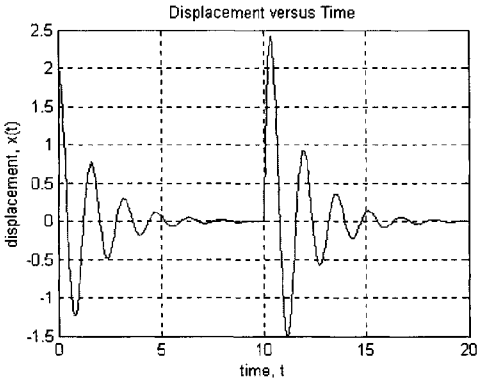


FIGURE 10.3.11

10.3.6 Solution by Superposition

Consider a 1-DOF system described by

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F_1(t) + F_2(t) + \cdots + F_N(t), \quad t \geq 0$$
$$x(0) = x_0, \quad \dot{x}(0) = v_0 \tag{10.45}$$

where $F_j(t)$, $j = 1, 2, \ldots, N$, are external loads. By the principle of superposition, the total response of the system is

$$x(t) = x_{Free}(t) + x_1(t) + x_2(t) + \cdots + x_N(t) \tag{10.46}$$

where $x_{Free}(t)$ is the solution of Eq. (10.12), which is the response excited by the initial disturbances (x_0, v_0); and $x_j(t)$ is the solution of

$$\begin{gathered} m\ddot{x}_j(t) + c\dot{x}_j(t) + kx_j(t) = F_j(t), \quad t \geq 0 \\ x_j(0) = 0, \quad \dot{x}_j(0) = 0 \end{gathered} \tag{10.47}$$

which is the response excited by the jth load $F_j(t)$. The vibration problem of multiple loads can be solved by the Toolbox according to Window 3.10.

Window 3.10. Determination of Response to Multiple External Loads

Disturbances:

Initial disturbances $x(0) = x_0$, $\dot{x}(0) = v_0$
Multiple external loads $F_1(t)$, $F_2(t)$, ..., $F_N(t)$

Solution Method 1:

```
y = freevib(x0, v0) + forcedvib (Load1) + ··· + forcedvib (LoadN);
plotvib(y)
```

Here function `plotvib` is given in Window 3.12.

Solution Method 2:

```
[y0, t] = freevib(x0, v0);
y_total = y0 + forcedvib (Load1) + ··· + forcedvib (LoadN);
% plotting displacement
plot(t, y_total(1,:))
% plotting velocity
plot(t, y_total(2,:))
```

EXAMPLE 3.8

An underdamped system of parameters $m = 1$, $\omega_n = 5$, and $\xi = 0.2$ is subject to a half-cycle sine load shown in Fig. 10.3.12. The forcing function can be written as

$$F(t) = \sin(2t) + \sin(2(t - \pi/2))\, u(t - \pi/2)$$

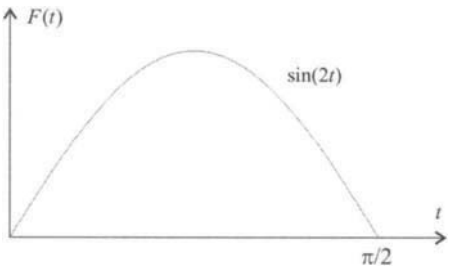


FIGURE 10.3.12

where $u(t-\pi/2)$ is the delayed unite step function. Let the system be at rest initially. By

```
>> setldof2(5, 0.2)
>> ld1 = ([4 1 2 0];
>> ld2 = [4 1 2 0 pi/2];
>> y = forcedvib(ld1) + forcedvib(ld2);
>> plotvib(y)
```

the time history of the displacement of the system is plotted in Fig. 10.3.13.

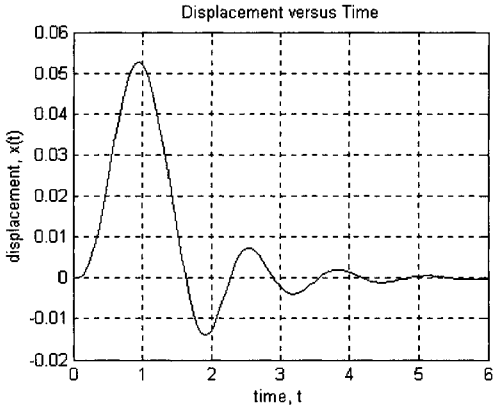


FIGURE 10.3.13

10.3.7 Mechanical Energy

The mechanical energy $E(t)$ of a 1-DOF system is the sum of the kinetic energy $U(t) = \frac{1}{2}m\dot{x}^2$ and potential energy $V(t) = \frac{1}{2}kx^2$; namely,

$$E(t) = U(t) + V(t) = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}kx^2 \tag{10.48}$$

Temporal integration of the equation of motion (10.41)

$$\int_0^t (m\ddot{x} + c\dot{x} + kx)\dot{x}dt = \int_0^t F\dot{x}dt$$

yields the work-energy principle

$$E(t) - E(0) = \int_0^t (F - c\dot{x})\dot{x}dt \tag{10.49}$$

where $E(0) = \frac{1}{2}mv_0^2 + \frac{1}{2}kx_0^2$ is the mechanical energy at the initial time $t = 0$. Equation (10.49) implies that the change of the energy is equal to the work done by the external force $F(t)$ and the damping force $-c\dot{x}$.

Differentiation of Eq. (10.49) gives the rate of change of the mechanical energy

$$\frac{d}{dt}E(t) = (m\ddot{x} + kx)\,\dot{x} = -c\dot{x}^2 + F\dot{x} \tag{10.50}$$

which is the power of the resultant of the external force and damping force. For an undamped system ($c = 0$) in free vibration, $dE(t)/dt = 0$, indicating that the energy is conserved, i.e., $E(t) = E(0)$ for any $t > 0$. For a damped system in free vibration, $dE(t)/dt = -c\dot{x}^2 \leq 0$, indicating that the mechanical energy is dissipated during vibration.

The Toolbox of this chapter provides a function `energy` for plotting time histories of the mechanical energy, kinetic energy, and potential energy of a system subject to initial and external disturbances; see Window 3.11.

Window 3.11. Function energy

MATLAB Function: **energy**

Purpose:
To plot the mechanical energy E, kinetic energy U, and potential energy V versus time.

Synopsis:

```
energy(y, t)
[E, U, V] = energy(y, t)
```

Description:
`energy(y, t)` plots the mechanical energy E, kinetic energy U, and potential energy V of a system versus time. Here `y` is a matrix containing the system response (displacement and velocity), and `t` is a vector containing the times of simulation. Both `y` and `t` can be obtained by functions `compltvib`, `freevib`, and `forcedvib`.

`[E, U, V] = energy(y, t)` returns the computed mechanical energy in a vector `E`, the kinetic energy in a vector `U`, and the potential energy in a vector `V`. With these output vectors, the energy functions can be plotted by the MATLAB function `plot`. For instance, `plot(t, U)` plots the kinetic energy of the system.

EXAMPLE 3.9

Consider a system with $m = 2$, $c = 0.5$, and $k = 2$. The system is subject to the initial conditions $x(0) = 2$ and $\dot{x}(0) = 0$, and a sinusoidal force $F(t) = 3 \sin 1.5t$.

```
>> set1dof(2, 0.5, 2)
>> [y, t] = compltvib(2, 0, [4 3 1.5 0], 50, 5001);
>> energy(y, t);
```

plots the displacement and mechanical energy of the system in Fig. 10.3.14, and the kinetic and potential energy in Fig. 10.3.15. As can be seen from Fig. 10.3.14, the total energy at steady state is sinusoidal with a frequency twice as high as the excitation frequency.

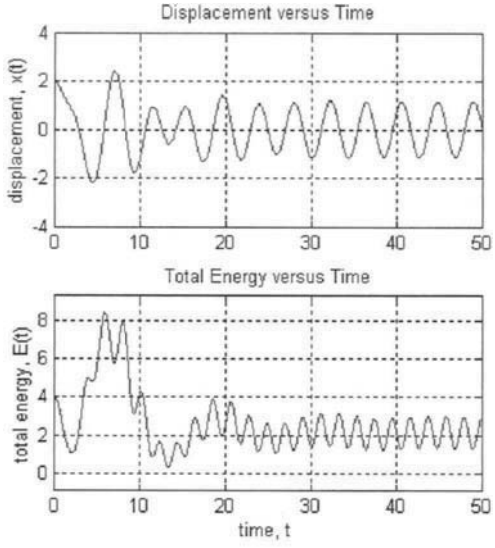


FIGURE 10.3.14

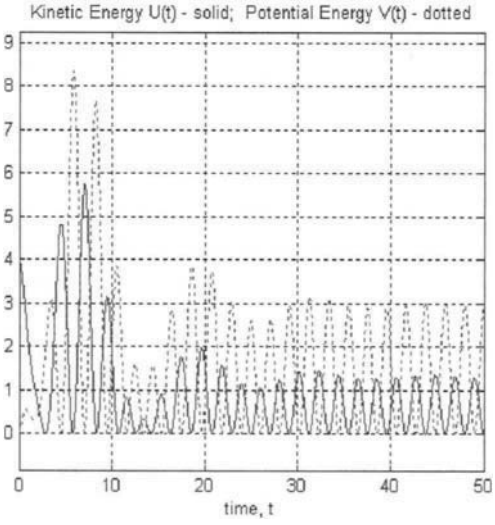


FIGURE 10.3.15

Now consider the free vibration of the system that is excited by the initial disturbances $x(0) = 2$ and $\dot{x}(0) = 0$.

```
>> [y, t] = freevib(2, 0, 50, 5001);
>> energy(y, t)
```

yields Fig. 10.3.16, which shows a monotonic decay in the mechanical energy.

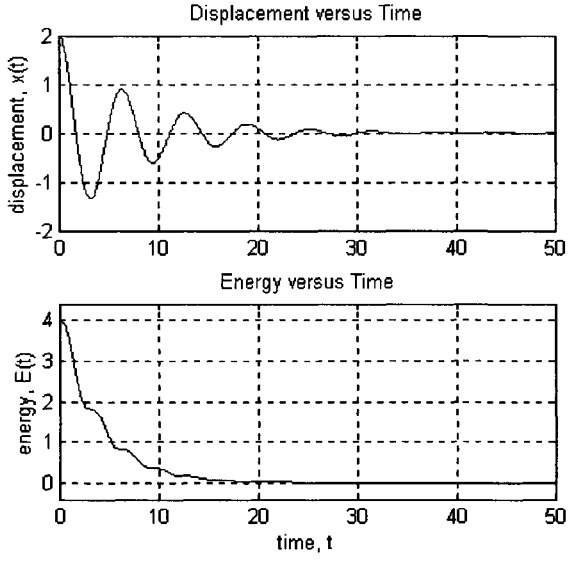


FIGURE 10.3.16

10.3.8 Plotting Computed Response

The Toolbox of this chapter has a function `plotvib` for plotting time history of dynamic response of vibrating systems; see Window 3.12. This function is often called after functions `freevib`, `forcedvib`, and `compltvib` have been executed for vibration analysis.

Window 3.12. Function `plotvib`

MATLAB Function: **`plotvib`**

Purpose:
To plot vibration response versus time and to display maximum response.

Synopsis:

```
plotvib(y)
plotvib(y, opt)
```

Description:
`plotvib(y)` plots the displacement of a vibrating system. Here, `y` is a matrix of computed response (displacement and velocity) that is obtained by function `freevib` or `forcedvib` or `compltvib`.

Window 3.12. Function `plotvib` (*Continued*)

`plotvib(y, opt)` plots system response with the following options:

`opt` = 0	plot displacement only
`opt` = 1	plot velocity only
`opt` = 2	plot both displacement and velocity in two subplots
`opt` = 3	plot both displacement and velocity in one plot

`plotvib` also displays maximum displacement and velocity, and the corresponding times.

EXAMPLE 3.10

Consider the vibrating system

$$\ddot{x}(t) + 10\dot{x}(t) + 1{,}000x(t) = 1.5\cos 3t + 50\sin 3t$$
$$x(0) = 0.01, \quad \dot{x}(0) = 3$$

```
>> setldof(1, 10, 1000)
>> y = freevib(0.01, 3,5, 501) + forcedvib([4 1.5 3 90], 5, 501)...
       + forcedvib([4 50 3 0], 5, 501);
>> plotvib(y, 2)
```

yields the displacement and velocity plots in Fig. 10.3.17, and gives the maximum displacement and velocity as follows:

```
Max displacement = 0.080723 at t = 0.04
Max velocity = 3 at t = 0
```

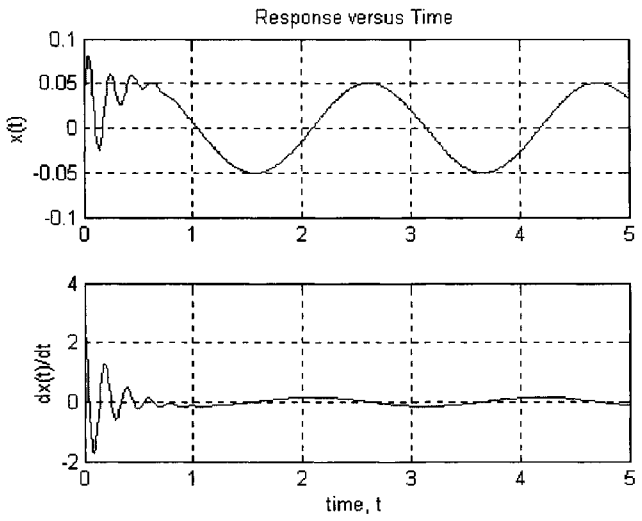


FIGURE 10.3.17

Sometimes, the user wants to use the standard MATLAB plotting function `plot`, to customize labels, titles, and other figure features. To this end, the times at which system response is computed should be obtained. Function `getpts` in Window 3.13 can be used to get access to the times of computation.

Window 3.13. Function getpts

MATLAB Function: **getpts**

Purpose:
To output the times at which the response of a vibrating system is computed.

Synopsis:

```
t = getpts
```

Description:
`t = getpts` returns the times of computation for the response of a vibrating system in a vector `t`. With `t` obtained by `getpts`, the response can be plotted by the MATLAB function `plot`.

Example:

A vibrating system of parameters $m = 1$, $c = 0.2$, and $k = 4$ is subject to initial disturbances $x_0 = 0.3$ and $v_0 = 2$ and a step external force $F(t) = 5$. The displacement of the system can be plotted as follows:

```
>> setldof(1, 0.2, 4)
>> y = freevib(0.3, 2) + forced([1 5]);
>> t = getpts;
>> plot(t, y(1,:))
```

10.4 Analytical Vibration Solutions

10.4.1 Analytical Methods

This section is concerned with analytical solution of the forced vibration problem

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t), \quad t \geq 0 \tag{10.51a}$$

$$x(0) = 0, \quad \dot{x}(0) = 0 \tag{10.51b}$$

Note that zero initial conditions have been assumed, as the analytical expressions of free vibration have been given in Section 10.3.1. Three analytical solution methods are presented herein. For the convenience of discussion, the following parameters are defined

$$\sigma = \xi\omega_n, \quad \omega_d = \sqrt{1-\xi^2}\,\omega_n, \quad \beta = \sqrt{\xi^2-1}\,\omega_n. \tag{10.52}$$

Method 1. Convolution Integral (Duhamel Integral) In this method, the solution of Eqs. (10.51a) and (10.51b) is expressed by the convolution integral

$$x(t) = \int_0^t F(\tau)g(t-\tau)\,d\tau \tag{10.53}$$

where $g(t)$ is the impulse response of the system described by

$$\begin{gathered} m\ddot{g}(t) + c\dot{g}(t) + kg(t) = \delta(t) \\ g(0) = 0, \quad \dot{g}(0) = 0 \end{gathered} \tag{10.54}$$

with $\delta(t)$ being the delta function. The $g(t)$ is given in the following damping cases.

Case 1. Undamped systems ($\xi = 0$)

$$g(t) = \frac{1}{m\omega_n} \sin \omega_n t \tag{10.55a}$$

Case 2. Underdamped systems ($0 < \xi < 1$)

$$g(t) = \frac{1}{m\omega_d} e^{-\sigma t} \sin \omega_d t \tag{10.55b}$$

Case 3. Critically-damped systems ($\xi = 1$)

$$g(t) = \frac{1}{m}\, t\, e^{-\omega_n t} \tag{10.55c}$$

Case 4. Overdamped systems ($\xi > 1$)

$$g(t) = \frac{1}{m\beta} e^{-\sigma t} \sinh \beta t \tag{10.55d}$$

Method 2. Laplace Transform In this method, Eq. (10.51a) is first Laplace-transformed with respect to time, which yields

$$\left(ms^2 + cs + k\right) X(s) = F(s) \tag{10.56}$$

where s is the complex Laplace transform parameter; and $X(s)$ and $F(s)$ are the Laplace transforms of the displacement $x(t)$ and the forcing function $F(t)$, respectively. It follows that

$$X(s) = G(s)F(s) \tag{10.57}$$

with

$$G(s) = \frac{1}{ms^2 + cs + k} \tag{10.58}$$

The $G(s)$ is called the *transfer function* of the vibrating system. The forced response is thus determined by inverse Laplace transform

$$x(t) = \mathcal{L}^{-1}\left[G(s)F(s)\right] \tag{10.59}$$

where $\mathcal{L}^{-1}$ is the inverse Laplace transform operator.

The transfer function $G(s)$ and the impulse response $g(t)$ are a Laplace transform pair, namely,

$$G(s) = \mathcal{L}[g(t)] \tag{10.60}$$

In fact, it is easy to show that the Laplace transform of $g(t)$ in Eq. (10.54) yields the expression in Eq. (10.58). Also, by the convolution theorem, the inverse Laplace transform of Eq. (10.59) leads to the integral in Eq. (10.53).

Method 3. Method of Undetermined Coefficients The forced response governed by Eqs. (10.51) can be written as

$$x(t) = x_p(t) + x_h(t) \tag{10.61}$$

where $x_p(t)$ is a particular solution of Eq. (10.51a), and $x_h(t)$ is the solution of the complementary homogenous equation whose coefficients are such that the solution given by Eq. (10.61) satisfies the conditions (10.51b). Consider a forcing function of the general format

$$F(t) = e^{\alpha t}[A_l(t)\cos\omega t + B_m(t)\sin\omega t] \tag{10.62}$$

where $A_l(t)$ and $B_m(t)$ are polynomials of orders l and m, respectively. By the theory of differential equations, a particular solution of Eq. (10.51a) is

$$x_p(t) = t^{\mu}e^{\alpha t}[Q_n(t)\cos\omega t + R_n(t)\sin\omega t] \tag{10.63}$$

where $Q_n(t)$ and $R_n(t)$ are polynomials of order $n = \max(l, m)$

$$\begin{aligned} Q_n(t) &= q_n t^n + q_{n-1}t^{n-1} + \cdots + q_1 t + q_0 \\ R_n(t) &= r_n t^n + r_{n-1}t^{n-1} + \cdots + r_1 t + r_0 \end{aligned} \tag{10.64}$$

and

$$\mu = \begin{cases} 0, & \text{if } \alpha + j\omega \neq -\xi\omega_n + \sqrt{\xi^2 - 1}\,\omega_n \\ 1, & \text{if } \alpha + j\omega = -\xi\omega_n + \sqrt{\xi^2 - 1}\,\omega_n \end{cases} \tag{10.65}$$

with $j = \sqrt{-1}$. Here, ω_n and ξ are the natural frequency and damping ratio of the vibrating system. The coefficients q_k and r_k are determined by first substituting Eqs. (10.64) into Eq. (10.51a) and then matching the coefficients of the terms on both sides of the resulting equation.

For an undamped system ($\xi = 0$) subject to a forcing function

$$F(t) = A_l(t)\cos\omega t + B_m(t)\sin\omega t \tag{10.66}$$

a particular solution by Eq. (10.62) is

$$x_p(t) = t^{\mu}[Q_n(t)\cos\omega t + R_n(t)\sin\omega t] \tag{10.67}$$

with

$$\mu = \begin{cases} 0, & \text{if } \omega \neq \omega_n \\ 1, & \text{if } \omega = \omega_n \end{cases} \tag{10.68}$$

Here, $\mu = 1$ indicates an unbounded resonant vibration.

Once a particular solution is found, the solution of Eqs. (10.51) is obtained as follows:

(i) For an undamped system ($\xi = 0$)

$$x(t) = x_p(t) - x_p(0)\cos\omega_n t - \frac{1}{\omega_n}\dot{x}_p(0)\sin\omega_n t \tag{10.69a}$$

(ii) For an underdamped system ($0 < \xi < 1$)

$$x(t) = x_p(t) - e^{-\sigma t}\left(x_p(0)\cos\omega_d t + \frac{\dot{x}_p(0) + \sigma x_p(0)}{\omega_d}\sin\omega_d t\right) \tag{10.69b}$$

(iii) For a critically-damped system ($\xi = 1$)

$$x(t) = x_p(t) - e^{-\omega_n t}\left\{x_p(0) + \left(\dot{x}_p(0) + \omega_n x_p(0)\right)\cdot t\right\} \tag{10.69c}$$

(iv) For an overdamped system ($\xi > 1$)

$$x(t) = x_p(t) - e^{-\sigma t}\left(x_p(0)\cosh\beta t + \frac{\dot{x}_p(0) + \sigma x_p(0)}{\beta}\sinh\beta t\right) \tag{10.69d}$$

EXAMPLE 4.1

To demonstrate the above-mentioned solution methods, consider an undamped system subject to a sinusoidal force:

$$m\ddot{x}(t) + kx(t) = F_0\sin\omega_n t$$
$$x(0) = 0, \quad \dot{x}(0) = 0$$

where $\omega_n = \sqrt{k/m}$. Note that the excitation frequency is the same as the natural frequency, which renders a resonant vibration. By the convolution integral (10.53), the forced response is obtained as

$$x(t) = \frac{F_0}{m\omega_n}\int_0^t \sin\omega_n(t-\tau)\sin\omega_n\tau\cdot d\tau$$

$$= \frac{F_0}{m\omega_n}\int_0^t [\sin\omega_n t\cos\omega_n\tau - \cos\omega_n t\sin\omega_n\tau]\sin\omega_n\tau\cdot d\tau$$

$$= \frac{F_0}{m\omega_n}\left\{\frac{\sin\omega_n t}{\omega_n}\int_0^t \sin\omega_n\tau\cdot d(\sin\omega_n\tau) - \frac{\cos\omega_n t}{2}\int_0^t [1-\cos 2\omega_n\tau]\cdot d\tau\right\}$$

$$= \frac{F_0}{m\omega_n}\left\{\frac{\sin\omega_n t}{2\omega_n}\sin^2\omega_n\tau - \frac{\cos\omega_n t}{2}\left[\tau - \frac{1}{2\omega_n}\sin 2\omega_n\tau\right]\right\}_{\tau=0}^{\tau=t}$$

$$= \frac{F_0}{2k}(\sin\omega_n t - \omega_n t\cos\omega_n t)$$

where $m\omega_n^2 = k$ has been used. As can be seen, $x(t)$ grows unbounded as time t increases.

If Laplace transform is applied,

$$X(s) = \frac{1}{ms^2 + k}\frac{F_0\omega_n}{s^2 + \omega_n^2} = \frac{F_0\omega_n}{m}\frac{1}{\left(s^2 + \omega_n^2\right)^2}$$

By partial fraction expansion

$$\frac{1}{\left(s^2 + \omega_n^2\right)^2} = \frac{a_1}{s - j\omega_n} + \frac{a_2}{s + j\omega_n} + \frac{b_1}{(s - j\omega_n)^2} + \frac{b_2}{(s + j\omega_n)^2}$$

where $j = \sqrt{-1}$, and

$$a_1 = -j\frac{1}{4\omega_n^3},\quad a_2 = j\frac{1}{4\omega_n^3},\quad b_1 = b_2 = -\frac{1}{4\omega_n^2}.$$

It follows that

$$\begin{aligned} x(t) &= \mathcal{L}^{-1}[X(s)] \\ &= \frac{F_0\omega_n}{m}\left\{a_1 e^{j\omega_n t} + a_2 e^{-j\omega_n t} + t\left(b_1 e^{j\omega_n t} + b_2 e^{-j\omega_n t}\right)\right\} \\ &= \frac{F_0\omega_n}{m}\left\{\frac{1}{2\omega_n^3}\frac{e^{j\omega_n t} - e^{-j\omega_n t}}{2j} - \frac{t}{2\omega_n^2}\frac{e^{j\omega_n t} + e^{-j\omega_n t}}{2}\right\} \\ &= \frac{F_0}{2k}(\sin\omega_n t - \omega_n t\cos\omega_n t) \end{aligned}$$

By the method of undetermined coefficients, a particular solution is

$$x_p(t) = t(a\cos\omega_n t + b\sin\omega_n t)$$

where Eq. (10.67) has been used. Substituting $x_p(t)$ into the equations of motion gives

$$2m\omega_n(-a\sin\omega_n t + b\cos\omega_n t) = F_0\sin\omega_n t$$

Matching the coefficients of sine and cosine terms leads to

$$a = -\frac{F_0}{2m\omega_n},\quad b = 0$$

So, the particular solution is

$$x_p(t) = -\frac{F_0}{2k}\omega_n t\cos\omega_n t,$$

and by Eq. (10.69a) the forced response is obtained as

$$x(t) = -\frac{F_0}{2k}\omega_n t \cos \omega_n t + \frac{F_0}{2k} \sin \omega_n t.$$

10.4.2 Solution by the Toolbox

The Toolbox of this chapter provides a function `forcedANL` for obtaining exact mathematical expressions of the forced response governed by Eqs. (10.51); see Window 4.1.

Window 4.1. Function `forcedANL`

MATLAB Function: **`forcedANL`**

Purpose:
To obtain the mathematical expressions of the forced response of a 1-DOF system.

Synopsis:

```
forcedANL(Load_Spec)
ya = forcedANL(Load_Spec)
```

Description:
`forcedANL(Load_Spec)` displays the mathematical expressions of the forced response of a 1-DOF system subject to an external force specified by a vector or matrix `Load_Spec` according to Table 10.4.1. Execution of function `forcedANL` also yields a plot of system response versus time.

`ya = forcedANL(Load_Spec)` returns the coefficients of the mathematical expressions of the forced response in a three-dimensional matrix `ya`. With `ya`, the forced response can be plotted by function `plotANL`; see Window 4.2.

TABLE 10.4.1 Specification of Forcing Functions for Function `forcedANL`

External Load	*F(t)*	*Load_Spec*
(L0) Impulse	$I_0\delta(t)$	[0 I_0]
(L1) Step	F_0	[1 F_0]
(L2) Ramp	$a_0 t$	[2 a_0]
(L3) Exponential	Ae^{bt}	[3 A b]
(L4) Sinusoidal	$A \sin(\omega t + \phi_0)$	[4 A ω ϕ_0] ϕ_0 in degrees
(L5) General form	$e^{\alpha t}[A_n(t)\cos \omega t + B_n(t) \sin \omega t]$ with $A_n(t) = a_n t^n + \cdots + a_1 t + a_0$ $B_n(t) = b_n t^n + \cdots + b_1 t + b_0$ $a_n^2 + b_n^2 \neq 0$	$\begin{bmatrix} \alpha & a_n & \cdots & a_1 & a_0 \\ \omega & b_n & \cdots & b_1 & b_0 \end{bmatrix}$

TABLE 10.4.2 Vectors α_j and β_j and Associate Mathematical Expression $z_j(t)$

Vectors α_j and β_j	*Mathematical Expression*
[0 0 0 0 0 0]	$z_j(t) = 0$
[1 a 0 0 0 0]	$z_j(t) = a$
[2 a m 0 0 0]	$z_j(t) = a\,t^m$
[3 a σ 0 0 0]	$z_j(t) = a\,e^{\sigma t}$
[4 a σ m 0 0]	$z_j(t) = a\,t^m e^{\sigma t}$
[5 a b σ ω 0]	$z_j(t) = e^{\sigma t}(a\cos\omega t + b\sin\omega t)$
[6 a b σ ω m]	$z_j(t) = t^m e^{\sigma t}(a\cos\omega t + b\sin\omega t)$
[7 a b σ β 0]	$z_j(t) = e^{\sigma t}(a\cosh\beta t + b\sinh\beta t)$

Note 1. The matrix ya in Window 4.1 is an $n_L \times 6 \times 2$ matrix, with

$$\text{ya}(:,:,1) = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n_L} \end{bmatrix} \in R^{n_L \times 6}, \qquad \text{ya}(:,:,2) = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n_L} \end{bmatrix} \in R^{n_L \times 6} \tag{10.70}$$

describing displacement $x(t)$ and velocity $\dot{x}(t)$, respectively. Here n_L is the number of terms in a mathematical expression

$$z_1(t) + z_2(t) + \cdots + z_{nL}(t)$$

for displacement or velocity, and α_j or β_j are row vectors describing the jth term $z_j(t)$ in the expression. The entries and meaning of α_j and β_j are given in Table 10.4.2. For instance,

$$\text{ya(:,:,1)} = \begin{bmatrix} 1 & -0.25 & 0 & 0 & 0 & 0 \\ 5 & 0.264 & -0.178 & -0.359 & 6.785 & 0 \\ 4 & 0.867 & -0.051 & 1 & 0 & 0 \end{bmatrix}$$

indicates that the displacement is of the form

$$x(t) = -0.25 + e^{-0.359t}(0.264\cos 6.785t - 0.178\sin 6.785t) + 0.867te^{-0.051t}.$$

Note 2. The mathematical expressions expressed by the output argument ya of function `forcedANL` can be converted to digital data and plotted against time by function `plotANL`, as shown in Window 4.2.

Window 4.2. Function plotANL

MATLAB Function: **plotANL**

Purpose:
To plot a time history of the forced response of a 1-DOF system, according to an analytical solution.

Synopsis:

```
plotANL(ya)
plotANL(ya, opt)
plotANL(ya, opt, tf, npts, Td)
[y, t] = plotANL(ya, opt, tf, npts, Td)
```

Description:
`plotANL(ya)` plots a time history of the displacement of a 1-DOF system under an external force. Here, `ya` is a matrix describing the analytical expressions of a forced response that can be obtained by function `forcedANL`, such as `ya = forcedANL(Load_Spec)`.

`plotANL(ya, opt)` plots the forced response with the following options:

`opt` = 0	plot displacement only
`opt` = 1	plot velocity only
`opt` = 2	plot both displacement and velocity in two subplots
`opt` = 3	plot both displacement and velocity in one plot

`plotANL(ya, opt, tf, npts, Td)` plots the forced response with `tf` being the total time of computation, `npts` the number of time points in the region $0 \leq t \leq$ `tf`, and `Td` the shift (time delay) of the time history of the forced response. By default, `opt` = 0, `tf` = $8\pi/\omega_n$, `npts` = 501, and `Td` = 0. Any one of the parameters `opt`, `tf`, `npts`, and `Td` can be replaced by [ ] as a placeholder for its default value.

`[y, t] = plotANL(ya, opt, tf, npts, Td)` returns the computed forced response in a two-row matrix `y` and the times of computation in a vector `t`. Here, `y(1,:)` contains the displacement data, and `y(2,:)` contains the velocity data. With `y` and `t`, the time history of the forced response can be plotted by `plot(t, y(1,:))` for displacement and `plot(t, y(2,:))` for velocity, or by function `plotvib` from this Toolbox (see Window 3.12 of Section 10.3.8).

Note:
The time delay parameter `Td` allows `plotANL` to compute and display the response of a system under a delayed forcing function, as described in Section 10.3.5.

EXAMPLE 4.2

A vibrating system of parameters $m = 1$, $c = 0.5$, and $k = 4$ is subject to the external force $F(t) = 4(t + 1)\,e^{-0.2t}\cos(2t)$. The system is initially at rest. By

```
>> setldof(1, 0.5, 4)
>> forcedANL([-0.2 4 4; 2 0 0])
```

the mathematical expressions of the forced response is obtained as follows:

```
x(t) = Term 1 + Term 2 + ...
 with
 Term 1 = exp(-0.25*t)*[-204.1579*cos(1.9843*t) +
          (282.8395)*sin(1.9843*t)]
 Term 2 = exp(-0.2*t)*[204.1579*cos(2*t) + (-282.9728)*sin(2*t)]
 Term 3 = t*exp(-0.2*t)*[-5.5046*cos(2*t) + (18.3486)*sin(2*t)]
 (total 3 terms)

dx(t)/dt = Term 1 + Term 2 + ...
 with
 Term 1 = exp(-0.25*t)*[612.2818*cos(1.9843*t) +
          (334.4034)*sin(1.9843*t)]
 Term 2 = exp(-0.2*t)*[-612.2818*cos(2*t) + (-333.3726)*sin(2*t)]
 Term 3 = t*exp(-0.2*t)*[37.7982*cos(2*t) + (7.3394)*sin(2*t)]
 (total 3 terms)
```

Function `forcedANL` also yields a time history plot of the forced response in Fig. 10.4.1.

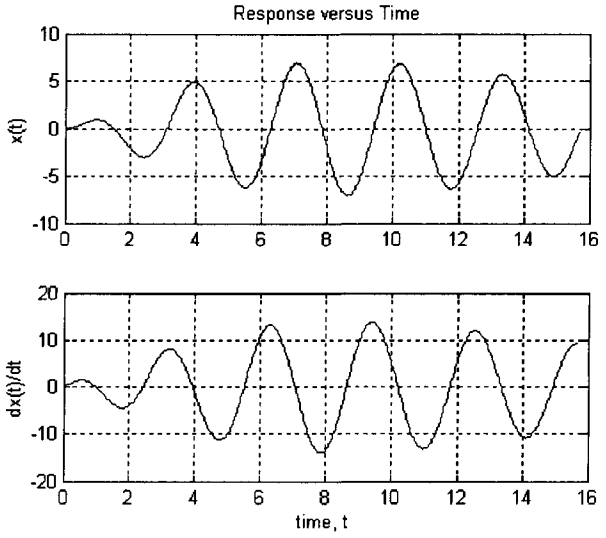


FIGURE 10.4.1

EXAMPLE 4.3

An undamped system is subject to a sequence of external forces:

$$m\ddot{x}(t) + kx(t) = 4\delta(t) - 4\delta(t - T) + 10(t - 2T)\, e^{-(t-2T)}u(t - 2T)$$

where $T = 2\pi\sqrt{m/k}$, $\delta(t)$ is the delta function, $u(t - T)$ and $u(t - 2T)$ are delayed unit step functions. Let the system parameters be $m = 1$ and $k = 4$. The delay parameter then is $T = \pi$. Assume that the system is at rest initially.

```
>> setldof(1, 0, 4)
>> ya = forcedANL([0 4]);
>> ya3 = forcedANL([-1 10 0; 0 0 0]);
```

yields the forced response of the system as follows:

$$x(t) = x_1(t) - x_1(t - T) \cdot u(t - T) + x_3(t - 2T) \cdot u(t - 2T)$$

where

$$x_1(t) = 2\sin(2t)$$

$$x_3(t) = 0.8e^{-t} + 2te^{-t} - 0.8\cos(2t) - 0.6\sin(2t).$$

To plot the above forced response, type the following in the MATLAB command window:

```
>> y1 = plotANL(ya,2);
>> y2 = plotANL(ya, 2,[ ], [ ], pi);
>> y3 = plotANL(ya3, 2,[ ], [ ], 2*pi);
>> y = y1-y2+y3;
>> plotvib(y,2)
```

which produces Fig. 10.4.2.

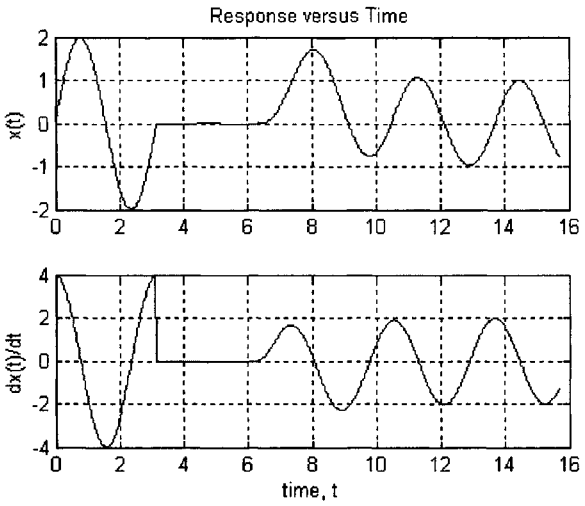


FIGURE 10.4.2

10.5 Frequency Response

In this section, we consider the steady-state response of a 1-DOF system to sinusoidal or harmonic excitation. Of special interest is the magnitude and phase of the response versus the excitation frequency. For this reason, steady-state response is also called *frequency response*.

10.5.1 Harmonic Excitation

In Fig. 10.5.1, a 1-DOF system mounted on a fixed foundation is subject to a harmonic excitation. The equation of motion of the system is

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F_0 \sin \omega t \tag{10.71}$$

The steady-state response of the system is

$$x_{ss}(t) = X \sin(\omega t + \phi) \tag{10.72}$$

where the magnitude X and phase angle ϕ are given by

$$X = \frac{F_0}{k} H(\omega), \qquad \phi = -\tan^{-1}\left(\frac{c\omega}{k - m\omega^2}\right) \tag{10.73}$$

with $H(\omega)$ being the amplification factor of the form

$$H(\omega) = \frac{X}{F_0/k} = \frac{1}{\sqrt{\left(1 - \omega^2/\omega_n^2\right)^2 + 4\xi^2\omega^2/\omega_n^2}} \tag{10.74}$$

Through introduction of frequency ratio $r = \omega/\omega_n$, the amplification factor is rewritten as

$$H(\omega) = \frac{1}{\sqrt{(1 - r^2)^2 + 4\xi^2 r^2}} \tag{10.75}$$

Also, for a harmonic excitation $F_0 \sin(\omega t + \theta_0)$, the steady-state response of the system is

$$x_{ss}(t) = X \sin(\omega t + \theta_0 + \phi) \tag{10.76}$$

with the magnitude X and phase angle ϕ being the same as given in Eq. (10.73).

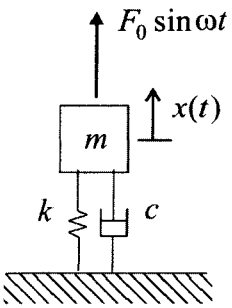


FIGURE 10.5.1 A system subject to a harmonic excitation.

For an underdamped system with damping ratio $\xi < \sqrt{2}/2$, the magnitude of its frequency response peaks at $\omega_p = \sqrt{1-2\xi^2}\,\omega_n$, with the maximum value

$$X_{\max} = \frac{F_0}{k}H(\omega_p) = \frac{F_0}{k}\cdot\frac{1}{2\xi\sqrt{1-\xi^2}} \tag{10.77}$$

For $\xi \geq \sqrt{2}/2$, the magnitude of the frequency response peaks at $\omega = 0$, with the maximum value F_0/k.

Force Transmissibility In design of structures and machines, the force applied to the foundation through the spring and damper is one concern. The total force transmitted to the foundation is

$$F_{trans}(t) = kx(t) + c\dot{x}(t) \tag{10.78}$$

which, at steady state, is given by

$$F_{trans}(t) = kX\sin(\omega t + \phi) + c\omega X\cos(\omega t + \phi) \tag{10.79}$$

The magnitude of the transmitted force is

$$F_T = X\sqrt{k^2 + c^2\omega^2} = \frac{F_0}{k}\sqrt{k^2 + c^2\omega^2}H(\omega) \tag{10.80}$$

The force transmissibility is defined as the ratio of the magnitude of the transmitted force to that of the applied force, namely,

$$T_f = \frac{F_T}{F_0} = \frac{1}{k}\sqrt{k^2 + c^2\omega^2}H(\omega) = \frac{\sqrt{1+4\xi^2r^2}}{\sqrt{(1-r^2)^2 + 4\xi^2r^2}} \tag{10.81}$$

where $r = \omega/\omega_n$. The force transmissibility T_f is used in vibration isolation of structures and machines; see Section 10.5.4.

Solution by the Toolbox To determine the steady-state magnitude and phase of a system under a harmonic excitation, function `harmonic` from the Toolbox can be used; see Window 5.1. Also, function `plotfr` in Window 5.2 can be used to plot the computed frequency response.

Window 5.1. Function `harmonic`

MATLAB Function: **`harmonic`**

Purpose:
To plot the steady-state displacement of a 1-DOF system subject to a harmonic excitation.

Window 5.1. Function `harmonic` (*Continued*)

Synopsis:

```
harmonic(f0)
harmonic
harmonic(f0, omgf, npts)
[y, Freq] = harmonic(f0, omgf, npts)
```

Description:

`harmonic(f0)` plots the magnitude and phase of the steady-state displacement of a 1-DOF system subject to a harmonic excitation of magnitude `f0`, versus the excitation frequency.

`harmonic` plots the system frequency response for `f0` = 1.

`harmonic(f0, omgf, npts)` plots the system frequency response in a frequency region $0 \leq \omega \leq$ `omgf`, where `npts` is the number of equally-spaced points in the frequency region for computation. By default, `omgf` is about three times the natural frequency ω_n of the system, and `npts` = 101. Any of the parameters `f0`, `omgf`, and `npts` can be replaced by [ ] as a placeholder for its default value.

`[y, Freq] = harmonic(f0, omgf, npts)` returns the computed frequency response in a two-row matrix `y`, in which the first row `y(1,:)` contains the magnitude and the second row `y(2,:)` contains the phase, and the frequencies of computation in a vector `Freq`. With `y` and `Freq`, the frequency response can be plotted by the MATLAB function `plot` as follows:

```
plot(Freq, y(1,:))    for magnitude
plot(Freq, y(2,:))    for phase
```

Also, the frequency response can be plotted by function `plotfr`; see Window 5.2.

Window 5.2. Function `plotfr`

MATLAB Function: **`plotfr`**

Purpose:

To plot the magnitude and phase of the frequency response of a 1-DOF system.

Synopsis:

```
plotfr(y)
plotfr(y, opt)
```

Description:

`plotfr(y)` plots the magnitude and phase of the frequency response of a system versus excitation frequency, where `y` is a matrix of frequency response data that is obtained by one of functions `harmonic`, `basevib`, and `unbalance`.

`plotfr(y, opt)` plots the frequency response with the following options:

`opt` = 0	plot both magnitude and phase
`opt` = 1	plot magnitude only
`opt` = 2	plot phase only

EXAMPLE 5.1

An underdamped system with parameters $m = 3$, $\omega_n = 10$, and $\xi = 0.15$ is subject to a harmonic excitation of magnitude $F_0 = 2$. By

```
>> set1dof2(10, 0.15, 3)
>> y = harmonic(2);
```

the frequency response of the system is plotted in Fig. 10.5.2. Also

```
>> plotfr(y,1)
```

yields the plot of the steady-state magnitude in Fig. 10.5.3.

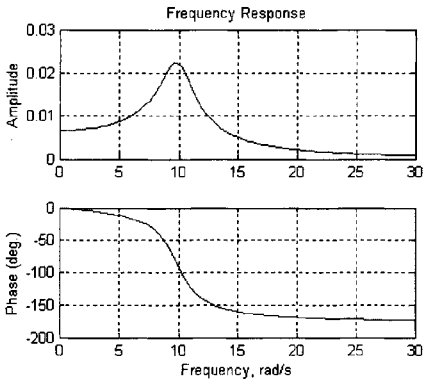


FIGURE 10.5.2

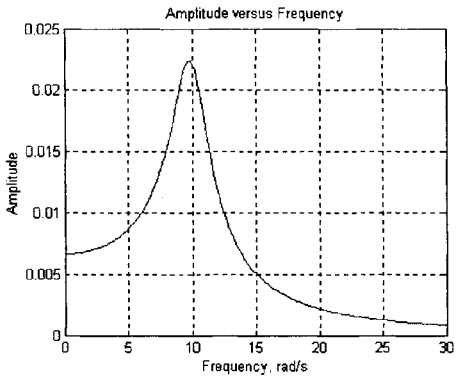


FIGURE 10.5.3

The Toolbox of this chapter provides a function `normhar` for plotting the amplification factor $H(\omega)$ and phase ϕ of the frequency response of a vibrating system, and a function `transm` for plotting force transmissibility T_f by Eq. (10.81); see Windows 5.3 and 5.4.

Window 5.3. Function normhar

MATLAB Function: **normhar**

Purpose:
To plot the amplification factor H and phase ϕ of the frequency response of a 1-DOF vibrating system.

Synopsis:

```
normhar
normhar(dmp_ratios)
normhar(dmp_ratios, opt, rf, npts)
```

Description:
`normhar` plots of the amplification factor and phase of system frequency response to harmonic excitation at the values of damping ratio $\xi = 0.05, 0.1, 0.2, 0.5, 0.7, 1.0$.

`normhar(dmp_ratios)` plots the amplification factor and phase of the frequency response at user-specified damping ratios, which are stored in a vector `dmp_ratios`, and displays the peak of each magnitude-frequency curve.

`normhar(dmp_ratios, opt, rf)` plots frequency response with the following options:

`opt` = 0 plot both magnitude and phase
`opt` = 1 plot magnitude only
`opt` = 2 plot phase only

Here `rf` defines a region of the frequency ratio $r = \omega/\omega_n$, namely, $0 \leq r \leq$ `rf`; and `npts` is the number of equally-spaced points in the frequency region for computation. By default, `rf` = 3, and `npts` = 301. Any of the arguments `dmp_ratios`, `opt`, and `rf` can be replaced by [ ] as a placeholder for its default value

Window 5.4. Function transm

MATLAB Function: **transm**

Purpose:
To plot force or displacement transmissibility of a 1-DOF vibrating system subject to harmonic excitation.

Synopsis:

```
transm(tr_ID)
transm(tr_ID, dmp_ratios, rf, npts)
[TR, freq_ratio] = transm(tr_ID, dmp_ratios, rf, npts)
```

Description:
`transm(tr_ID)` plots force or displacement transmissibility of a 1-DOF system at the damping ratios $\xi = 0.05, 0.1, 0.2, 0.5, 0.7, 1.0$, with the following options:

`tr_ID` = 1 plot force transmissibility due to external harmonic excitation

Window 5.4. Function `transm` (*Continued*)

`tr_ID` = 2 plot displacement transmissibility due to base excitation
`tr_ID` = 3 plot force transmissibility due to base excitation
`tr_ID` = 4 plot force transmissibility due to rotating unbalance

`transm(tr_ID, dmp_ratios, rf, npts)` plots force or displacement transmissibility, at user-specified damping ratios stored in a vector `dmp_ratios`, where `rf` defines a frequency region of simulation $0 \leq r \leq$ `rf`, with $r = \omega/\omega_n$, and `npts` is the number of equally-spaced points in the frequency region. Parameters `rf` and `dmp_ratios` can be replaced by [ ] as a placeholder. By default, `rf` = 3, and `npts` = 301.

`[TR, freq_ratio] = transm(tr_ID, dmp_ratios, rf, npts)` returns the computed transmissibility in a matrix `TR`, and frequencies of computation in a vector `freq_ratio`. The *j*th row of the matrix, `TR(j,:)`, contains the data of the transmissibility at the *j*th value of damping ratio from the input vector `dmp_ratios`.

EXAMPLE 5.2

Consider a vibrating system with $m = 4$ and $k = 100$. By

```
>> set1dof(4, 0, 100)
>> normhar([0.1 0.2 0.4 0.6 0.8 1.0 1.2])
```

the amplification factor and phase of the system frequency response are plotted in Fig. 10.5.4 at the damping ratios ξ = 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, and 1.2. Also,

```
>> transm(1, [0.1 0.2 0.4 0.6 0.8 1.0 1.2])
```

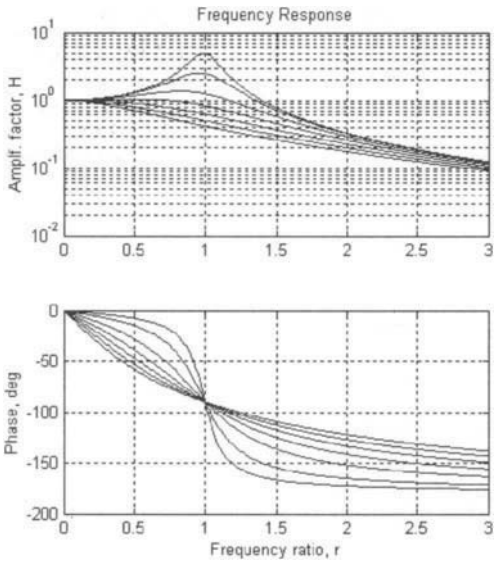


FIGURE 10.5.4

plots the force transmissibility in Fig. 10.5.5 at the specified damping ratios.

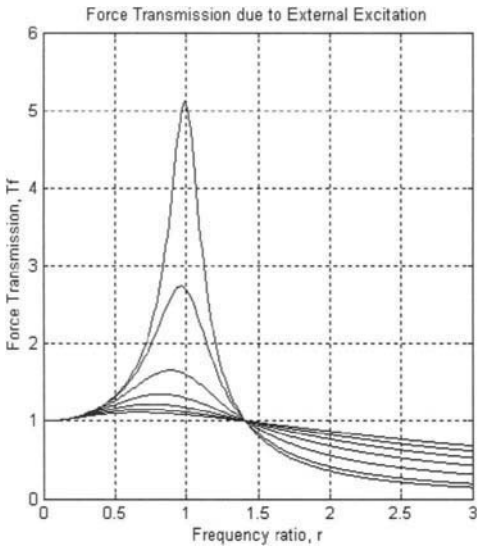


FIGURE 10.5.5

10.5.2 Base Excitation

In Fig. 10.5.6, a mass-damper-spring system is subject to a harmonic base (foundation) excitation, $y(t) = Y \sin \omega t$. The equation of motion of the system is

$$m\ddot{x}(t) = -c\,(\dot{x}(t) - \dot{y}(t)) - k\,(x(t) - y(t)) \tag{10.82}$$

or

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = Y\,(c\omega \cos \omega t + k \sin \omega t) \tag{10.83}$$

The transient response of the system can be obtained by functions `freevib` and `forcedvib`; see Section 10.3. The steady-state response of the system is

$$x_{ss}(t) = X \sin(\omega t + \phi) \tag{10.84}$$

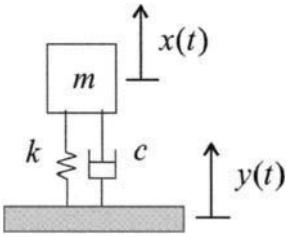


FIGURE 10.5.6 A system subject to a harmonic base excitation.

where the magnitude and phase are

$$X = Y\frac{\sqrt{k^2 + (c\omega)^2}}{\sqrt{(k - \omega^2 m)^2 + c^2\omega^2}} \tag{10.85}$$

$$\phi = \tan^{-1}\left(\frac{c\omega}{k}\right) - \tan^{-1}\left(\frac{c\omega}{k - m\omega^2}\right) \tag{10.86}$$

The steady-state magnitude peaks at the frequency

$$\omega_p = \frac{\omega_n}{2\xi}\sqrt{\sqrt{1 + 8\xi^2} - 1} \tag{10.87}$$

with the maximum value

$$X_{\max} = X|_{\omega=\omega_p} = Y\frac{\sqrt{k^2 + (c\omega_p)^2}}{\sqrt{(k - \omega_p^2 m)^2 + c^2\omega_p^2}}. \tag{10.88}$$

Displacement Transmissibility The displacement transmissibility T_d is a ratio of the magnitude of the displacement of the mass to that of the base excitation; that is,

$$T_d = \frac{X}{Y} = \frac{\sqrt{k^2 + (c\omega)^2}}{\sqrt{(k - \omega^2 m)^2 + c^2\omega^2}} = \frac{\sqrt{1 + 4\xi^2 r^2}}{\sqrt{(1 - r^2)^2 + 4\xi^2 r^2}} \tag{10.89}$$

where $r = \omega/\omega_n$ is the frequency ratio. The displacement transmissibility is used in vibration isolation of structures and machines; see Section 10.5.4.

Force Transmissibility The total force transmitted to the vibrating mass that is caused by the base excitation is

$$\begin{aligned} F_{trans}(t) &= -k\,(x(t) - y(t)) - c\,(\dot{x}(t) - \dot{y}(t)) \\ &= m\ddot{x}(t) = -m\omega^2 X\sin(\omega t + \phi) \end{aligned} \tag{10.90}$$

where Eq. (10.82) has been used. The force transmissibility is a ratio of the magnitude of the transmitted force to kY, where Y is the amplitude of the base excitation. Thus

$$T_f = \frac{F_T}{kY} = \frac{m\omega^2 X}{kY} = \frac{r^2\sqrt{1 + 4\xi^2 r^2}}{\sqrt{(1 - r^2)^2 + 4\xi^2 r^2}} \tag{10.91}$$

Solution by the Toolbox This Toolbox has a function `basevib` for computing frequency response to base excitation; see Window 5.5.

Window 5.5. Function basevib

MATLAB Function: **basevib**

Purpose:
To plot the steady-state response of a 1-DOF system under harmonic base excitation.

Synopsis:

```
basevib(Y)
basevib(Y, omgf, npts)
[X, Freq] = basevib(Y, omgf, npts)
```

Description:
`basevib(Y)` plots the steady-state magnitude and phase of a 1-DOF system subject to a harmonic base excitation of magnitude `Y`.

`basevib(Y, omgf, npts)` plots the steady-state magnitude and phase in a frequency region $0 \leq \omega \leq$ `omgf`, where `npts` is the number of points selected in the frequency region for computation. By default, `omgf` is about three times the natural frequency ω_n, and `npts` = 101. Any of the parameters `Y`, `omgf`, and `npts` can be replaced by [ ] as a placeholder for its default value.

`[X, Freq] = basevib(Y, omgf, npts)` returns the frequency response in a two-row matrix `X` and the frequencies of simulation in a vector `Freq`. With `X` and `Freq`, the steady-state magnitude and phase can be plotted by the MATLAB function `plot` as follows:

```
plot(Freq, X(1,:))    for magnitude
plot(Freq, X(2,:))    for phase
```

Also, function `plotfr` can be used to plot the frequency response; see Window 5.2.

In addition, function `transm` in Window 5.4 can be used to plot the transmissibilities as follows:

(a) For the displacement transmissibility T_d given in Eq. (10.89), type

```
>> transm(2, dmp_ratios)
```

(b) For the force transmissibility T_f given in Eq. (10.91), use

```
>> transm(3, dmp_ratios)
```

EXAMPLE 5.3

A system of parameters $m = 2$, $c = 2$, and $k = 15$ is subject to a harmonic base excitation of magnitude `Y` = 3.5. The steady-state response of the system is plotted in Fig. 10.5.7 by

```
>> set1dof(2, 2, 15)
>> basevib(3.5,10)
```

The peak of the magnitude curve is determined as $X_{max} = 10.34$ at $\omega = 2.66$ rad/s. Also, to plot the displacement transmissibility at damping ratios $\xi = 0.3, 0.6, 0.9, 1.2$, execute

```
>> transm(2, [0.3 0.6 0.9 1.2])
```

which yields Fig. 10.5.8.

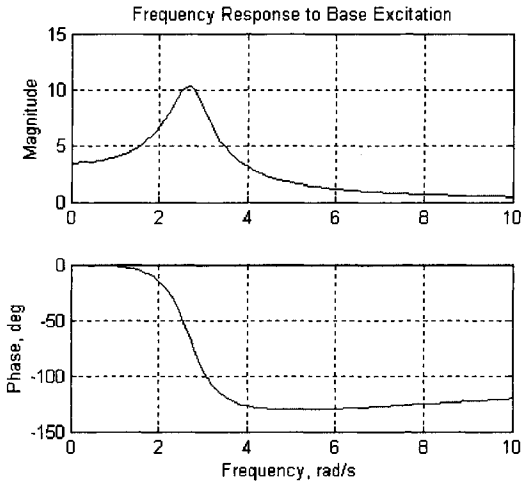


FIGURE 10.5.7

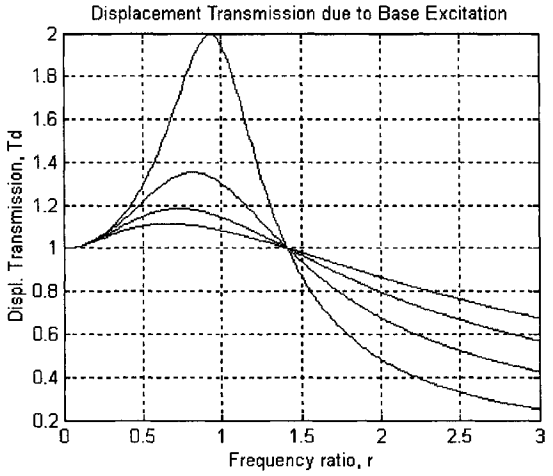


FIGURE 10.5.8

10.5.3 Response under Rotating Unbalance

A schematic of rotating unbalance that is often encountered in rotating machinery is shown in Fig. 10.5.9, where the machine with total mass m and rotation speed ω has an unbalanced mass

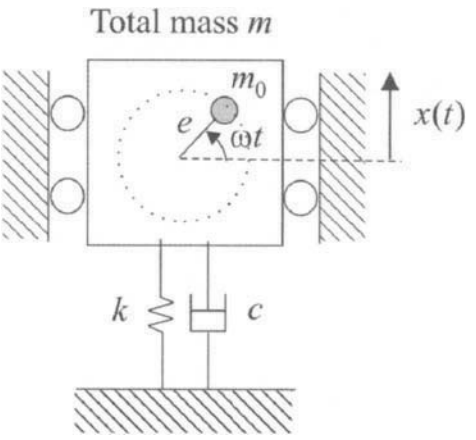


FIGURE 10.5.9 Rotating unbalance.

m_0 at a distance e from the rotation center. The support or mount of the machine is modeled as a spring and damper. Assume that the machine is constrained by vertical friction-free walls such that a horizontal motion is not possible. The vertical motion $x(t)$ of the rotation center of the machine is governed by (References 3 and 5)

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = m_0 e\omega^2 \sin \omega t \tag{10.92}$$

The steady-state solution of Eq. (10.92) is

$$x_{ss}(t) = X \sin(\omega t + \phi) \tag{10.93}$$

where the magnitude and phase are

$$\begin{aligned} X &= \frac{m_0\, e\, \omega^2}{\sqrt{(k - m\omega^2)^2 + c^2\omega^2}} = \frac{m_0 e}{m} \frac{r^2}{\sqrt{(1 - r^2)^2 + 4\xi^2 r^2}} \\ \phi &= -\tan^{-1}\left(\frac{c\omega}{k - m\omega^2}\right) = -\tan^{-1}\left(\frac{2\xi r}{1 - r^2}\right) \end{aligned} \tag{10.94}$$

with $r = \omega/\omega_n$. For $0 < \xi < \sqrt{2}/2$, the magnitude peaks at $\omega_p = \omega_n/\sqrt{1 - 2\xi^2}$, with the value

$$X_{\max} = \frac{m_0 e}{m} \frac{1}{2\xi\sqrt{1 - \xi^2}} \tag{10.95}$$

For $\xi > \sqrt{2}/2$, the magnitude does not have a maximum at a finite frequency, and grows from zero at $\omega = 0$ to 1 as $\omega \to \infty$.

Force Transmissibility The force transmitted to the foundation due to a rotating unbalance is

$$F_{trans}(t) = kX \sin(\omega t + \phi) + c\omega X \cos(\omega t + \phi) \tag{10.96}$$

where X is given in Eq. (10.94). The magnitude of the transmitted force is $F_T = X\sqrt{k^2 + c^2\omega^2}$. The force transmissibility T_f is the ratio of the magnitude of the transmitted force to that of the excitation force caused by the rotating unbalance, namely,

$$T_f = \frac{F_T}{m_0 e\omega^2} = \frac{\sqrt{1 + 4\xi^2 r^2}}{\sqrt{(1 - r^2)^2 + 4\xi^2 r^2}} \tag{10.97}$$

Note that the force transmissibility here is the same as given in Eq. (10.81). The force transmissibility is used in vibration isolation of structures and machines; see Section 10.5.4.

Solution by the Toolbox The Toolbox of this chapter provides a function `unbalance` for computing unbalanced steady-state response; see Window 5.6. Also, the force transmissibility T_f defined in Eq. (10.97) can be plotted by function `transm` given in Window 5.4.

Window 5.6. Function unbalance

MATLAB Function: **unbalance**

Purpose:
To plot the steady-state response of a rotating machine with an unbalanced mass.

Synopsis:

```
unbalance(m0, e)
unbalance(m0, e, omgf, npts)
[y, Freq] = unbalance(m0, e, omgf, npts)
```

Description:
`unbalance(m0, e)` plots the magnitude and phase of the steady-state displacement of a rotating machine with an unbalanced mass `m0` that is a distance `e` from the rotation center. Also, `unbalance` plots the steady-state response for `m0` = 1 and `e` = 1.

`unbalance(m0, e, omgf, npts)` plots the steady-state displacement in the frequency region $0 \le \omega \le$ `omgf`, where `npts` is the number of points in the frequency region. By default, `m0` = 1, `e` = 1, `omgf` is about $3\omega_n$, and `npts` = 101. Any of the parameters `m0`, `e`, `omgf`, and `npts` can be replaced by [ ] as a placeholder for default value.

`[y, Freq] = unbalance(m0, e, omgf, npts)` returns computed steady-state response in a matrix `y` and the frequencies of computation in a vector `Freq`. The first row `y(1,:)` of `y` contains the magnitude data; the second row `y(2,:)` contains the phase data. With `y` and `Freq`, the steady-state response can be plotted by the MATLAB function `plot` as follows:

```
plot(Freq, y(1,:))      for magnitude
plot(Freq, y(2,:))      for phase
```

Also, function `plotfr` can be used to plot the frequency response; see Window 5.2.

EXAMPLE 5.4

A rotor of parameters $m = 250$ kg, $k = 625{,}000$ N/m, and $\xi = 0.02$ has an unbalance of $m_0 e = 0.01$ kg-m. By

```
>> wn = sqrt(625000/250);
>> set1dof2(wn, 0.02, 250)
>> unbalance(0.01)
```

the frequency response of the rotor due to the unbalance is plotted in Fig. 10.5.10. The force transmissibility curves of the system are the same as those given in Fig. 10.5.5.

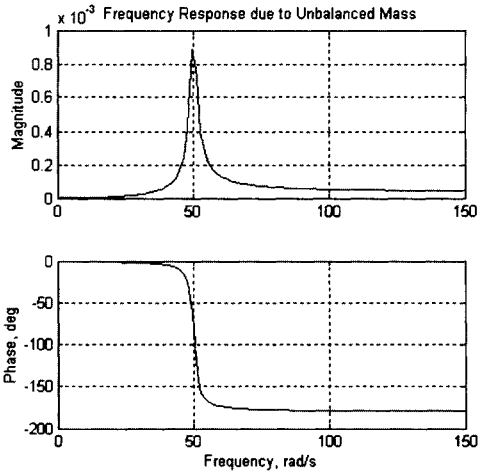


FIGURE 10.5.10

10.5.4 Vibration Isolation

Vibration isolation is a commonly used technique for reducing or suppressing unwanted vibrations in structures and machines. With this technique, the device or system of interest is isolated from the source of vibration through insertion of a resilient member or isolator. There are various types of isolators, including metal springs, rubber mounts, and pneumatic mounts. Vibration isolation is usually applied in the following two situations:

(a) The foundation of a vibrating system is protected from large transmitted forces due to harmonic excitation (Section 10.5.1) or rotating unbalanced mass (Section 10.5.3); and
(b) The vibrating system, which may be a delicate device or instrument, is protected from the motion of its base (Section 10.5.2).

In either situation, a transmissibility of the vibration isolation system is modified at a given frequency, so as to reduce forces transmitted to the foundation or to suppress displacement transmitted to the delicate device in consideration.

TABLE 10.5.1 Three Problems of Vibration Isolation

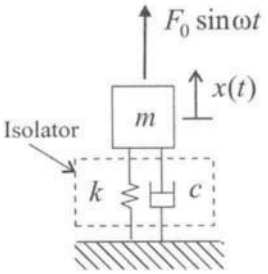
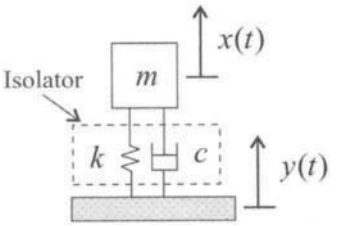
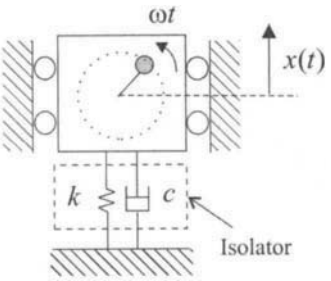
Problem 1. External Excitation (see Section 10.5.1)	*Problem 2. Base Excitation* (see Section 10.5.2)	*Problem 3. Rotating Unbalance* (see Section 10.5.3)
		
Source of vibration: $F(t) = F_0 \sin \omega t$ (external load)	Source of vibration: $y(t) = Y \sin \omega t$ (motion of the base)	Source of vibration: $m_0 e\omega^2 \sin \omega t$ (unbalanced mass)
What to protect: Foundation	What to protect: Vibrating mass	What to protect: Foundation
Force transmissibility: $T_f = \dfrac{\sqrt{1+4\xi^2 r^2}}{\sqrt{(1-r^2)^2+4\xi^2 r^2}}$	Displacement transmissibility: $T_d = \dfrac{\sqrt{1+4\xi^2 r^2}}{\sqrt{(1-r^2)^2+4\xi^2 r^2}}$	Force transmissibility: $T_f = \dfrac{\sqrt{1+4\xi^2 r^2}}{\sqrt{(1-r^2)^2+4\xi^2 r^2}}$
Guidance for isolator design: $T_f <$ desired value	Guidance for isolator design: $T_d <$ desired value	Guidance for isolator design: $T_f <$ desired value

Table 10.5.1 depicts three vibration isolation problems that are defined according to vibration sources, namely, external excitation, base excitation, and rotating unbalance. The isolators in these problems are modeled as a pair of spring and damper. The steady-state response and transmissibilities in these problems have been given in Sections 10.5.1 to 10.5.3. Note that T_f and T_d listed in Table 10.5.1 have the same mathematical expressions, although their physical indications are different. In each of these problems, the parameters c and k of an isolator are designed such that the force or displacement transmissibility is below a desired value.

The Toolbox of this chapter has a function `isolator` for selecting the parameters of an isolator for the three problems listed in Table 10.5.1; see Window 5.7. Also, function `transm` in Window 5.4 can be used to provide a guideline for vibration isolator design.

Window 5.7. Function `isolator`

MATLAB Function: **`isolator`**

Purpose:
To determine the spring and damping coefficients of a vibration isolator.

Window 5.7. Function `isolator` (*Continued*)

Synopsis:

```
isolator(prob_ID, TR, omega, m)
isolator(prob_ID, TR, omega, m, dmp_ratios)
[k, c, r] = isolator(prob_ID, TR, omega, m, dmp_ratios)
```

Description:

`isolator(prob_ID, TR, omega, m)` determines the spring coefficient `k` and damping coefficient `c` of a vibration isolator at the damping ratios $\xi = 0$, 0.01, 0.02, 0.05, 0.10, 0.15, 0.2, 0.3, and 0.5, where `TR` is the desired transmissibility value, `omega` is the excitation frequency in rad/s, `m` is the given mass of the vibration isolation system, and `prob_ID` is an integer indicating the problem of interest as follows:

`prob_ID` = 1 for Problem 1 (external excitation) in Table 10.5.1
`prob_ID` = 2 for Problem 2 (base excitation) in Table 10.5.2
`prob_ID` = 3 for Problem 3 (rotating unbalance) of Table 10.5.3

`isolator(prob_ID, TR, omega, m, dmp_ratios)` determines the spring and damping coefficients of an isolator at the damping ratios stored in a vector `dmp_ratios`.

`[k, c, r] = isolator(prob_ID, TR, omega, mass, dmp_ratios)` returns the values of the spring and damping coefficients of an isolator, and associated frequency ratios in vectors `k`, `c`, and `r`, respectively.

Given transmissibility (T_f or T_d), damping ratio ξ, and excitation frequency ω, function `isolator` first solves the following transmissibility equation

$$T_R^2\left((1-r^2)^2+4\xi^2r^2\right)-1-4\xi^2r^2=0 \tag{10.98}$$

for a frequency ratio r, where $T_R = T_f$ for Problems 1 and 3 and $T_R = T_d$ for Problem 2. Function `isolator` then determines the natural frequency of the isolator from $\omega_n = \omega/r$. With ω_n, the parameters c and k of the isolator are given by

$$c = 2m\xi\omega_n, \quad k = m\omega_n^2$$

where m is the mass of the vibration isolation system.

EXAMPLE 5.5

A rotor of mass $m = 250$ kg and rotation speed at 2,400 rpm is mounted on a fixed foundation. If only 20% of the unbalanced force is permitted to be transmitted to the foundation, what are the values of k and c of the mount (isolator)? Assume that the damping ratio $\xi = 0.07$.

Solution. This is Problem 3 in Table 10.5.1, with force transmissibility $T_f = 0.2$ and excitation frequency $\omega = 2\pi \times (2{,}400/60) = 80\pi$ rad/s. By

```
>> transm(4, 0.07)
```

the force transmissibility of the vibration isolation system is plotted in Fig. 10.5.11, which indicates that $T_f < 0.2$ if the frequency ratio r is approximately larger than 2.5. Also,

```
>> isolator(3, 0.2, 80*pi, 250, 0.07)
```

yields

$$k = 2.5116 \times 10^6 \text{ N/m}, \quad c = 3{,}508.1 \text{ kg/s}.$$

The frequency ratio in this case is found to be $r = \omega/\omega_n = 2.5075$.

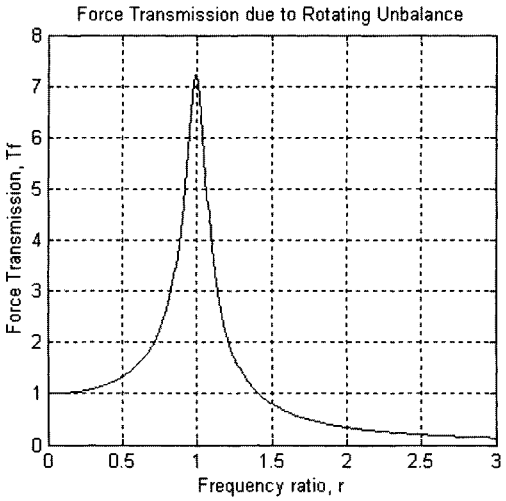


FIGURE 10.5.11

EXAMPLE 5.6

An instrument of mass $m = 5$ kg is mounted on a supporting base that experiences a motion $y(t) = 2\sin(23.5t)$ cm. Design a vibration isolator for the instrument such that its displacement is no large than 0.5 cm.

Solution. This is Problem 2 in Table 10.5.1. The displacement transmissibility of the vibration isolation system $T_d = X/Y = 0.25$. By

```
>> isolator(2, 0.25, 23.5, 5)
```

the spring and damping coefficients of the isolator at different values of the damping ratio are obtained; see the table below, by which an isolator of proper material may be selected.

Damping Ratio ξ	*Spring Coefficient k (N/m)*	*Damping Coefficient c (kg/s)*
0	552.25	0
0.01	551.836	1.0506
0.02	550.5964	2.0988
0.05	542.0161	5.2058
0.1	512.7346	10.1265
0.15	468.4316	14.5188
0.2	414.9628	18.22
0.3	304.8981	23.4268
0.5	154.7719	27.8183

10.6 Response to Periodic Excitation

In this section, the steady-state vibration of a 1-DOF system subject to a periodic force is obtained in Fourier series, which is an infinite sum of sine and cosine terms.

10.6.1 Steady-State Solution in Fourier Series

A 1-DOF system

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t) \tag{10.99}$$

is under a periodic external force $F(t)$ of period T. The periodic force, with property $F(t+T) = F(t)$ for any time t, can be expanded in a Fourier series

$$F(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos \frac{2k\pi}{T} t + b_k \sin \frac{2k\pi}{T} t \right) \tag{10.100}$$

where the coefficients

$$\begin{aligned} a_k &= \frac{2}{T} \int_0^T F(t) \cos \frac{2k\pi}{T} t dt, \quad k = 0, 1, 2, \ldots \\ b_k &= \frac{2}{T} \int_0^T F(t) \sin \frac{2k\pi}{T} t dt, \quad k = 1, 2, 3, \ldots \end{aligned} \tag{10.101}$$

The steady-state response of the system, by superposition, is written as

$$x_{ss}(t) = x_0(t) + \sum_{j=1}^{\infty} \left(x_j(t) + y_j(t) \right) \tag{10.102}$$

where x_j and y_j are the solutions of the following equations:

$$\begin{aligned} m\ddot{x}_0(t) + c\dot{x}_0(t) + kx_0(t) &= a_0/2 \\ m\ddot{x}_j(t) + c\dot{x}_j(t) + kx_j(t) &= a_j \cos j\omega t, \quad j = 1,2,\ldots \\ m\ddot{y}_j(t) + c\dot{y}_j(t) + ky_j(t) &= b_j \sin j\omega t, \quad j = 1,2,\ldots \end{aligned} \tag{10.103}$$

with

$$\omega = \frac{2\pi}{T} \tag{10.104}$$

If $c = 0$ and $\omega_n = l\omega$ for a nonzero integer l, resonant vibration occurs, and under the circumstances, steady-state solution does not exist.

If the vibration is nonresonant ($c \neq 0$ or $\omega_n \neq l\omega$), the solutions of Eq. (10.103) are obtained as follows:

$$\begin{aligned} x_0(t) &= \frac{a_0}{2k} \\ x_j(t) &= \frac{a_j}{k} h_j \cos\left(j\omega t + \phi_j\right), \quad j = 1,2,3,\ldots \\ y_j(t) &= \frac{b_j}{k} h_j \sin\left(j\omega t + \phi_j\right), \quad j = 1,2,3,\ldots \end{aligned} \tag{10.105}$$

where

$$\begin{aligned} h_j &= \frac{1}{\sqrt{(1 - j^2r^2)^2 + 4\xi^2 j^2 r^2}} \\ \phi_j &= -\tan^{-1}\left(\frac{cj\omega}{k - mj^2\omega^2}\right) = -\tan^{-1}\left(\frac{2\xi jr}{1 - j^2r^2}\right) \end{aligned} \tag{10.106}$$

with

$$r = \frac{\omega}{\omega_n} = \frac{2\pi}{T\omega_n} \tag{10.107}$$

In computation, the steady-state response is determined by a truncated series of $2n + 1$ terms

$$x_{ss}(t) \approx \frac{a_0}{2k} + \frac{1}{k}\sum_{j=1}^{n} h_j \left[a_j \cos\left(j\omega t + \phi_j\right) + b_j \sin\left(j\omega t + \phi_j\right)\right] \tag{10.108}$$

Sometimes, a periodic force is of irregular form; see Fig. 10.6.1, for instance. The Fourier coefficients in this case can be determined by numerical integration. To this end, divide the period T into N segments by the equally spaced points $t_i = i\,\Delta t$, $i = 0,1,2,\ldots,N$, with

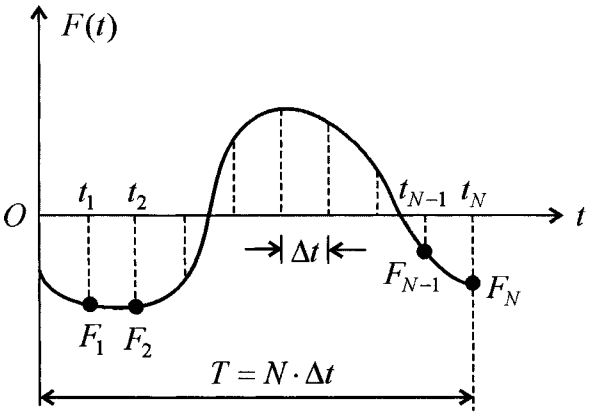


FIGURE 10.6.1 A periodic force of irregular form.

$\Delta t = T/N$. Application of the trapezoidal rule of numerical integration to Eq. (10.101) yields

$$a_0 = \frac{2}{N}\sum_{i=1}^{N} F_i$$

$$a_j = \frac{2}{N}\sum_{i=1}^{N} F_i \cos\frac{2j\pi t_i}{T}, \quad j = 1,2,\ldots \tag{10.109}$$

$$b_j = \frac{2}{N}\sum_{i=1}^{N} F_i \sin\frac{2j\pi t_i}{T}, \quad j = 1,2,\ldots$$

where $F_i = F(t_i)$. After these coefficients are obtained, formulas (10.105) to (10.108) can be used in vibration analysis.

10.6.2 Solution by the Toolbox

The Toolbox of this chapter provides a function `periodic` for computing the steady-state response of a 1-DOF system subject to a periodic external force; see Window 6.1.

Window 6.1. Function `periodic`

MATLAB Function: **`periodic`**

Purpose:
To plot the steady-state displacement of a 1-DOF system subject to a periodic external force.

Synopsis:

```
periodic(pLoad)
periodic(pLoad, n, tf, npts)
[y, t] = periodic(pLoad, n, tf, npts)
```

Window 6.1. Function periodic (*Continued*)

Description:
`periodic(pLoad)` plots the steady-state displacement of a 1-DOF system subject to an external periodic force specified by vector `pLoad` according to Table 10.6.1.

`periodic(pLoad, n, tf, npts)` plots the steady-state displacement by using the first `2n+1` terms of the Fourier series (10.108), where `tf` defines a time region $0 \leq t \leq$ `tf` for computation, and `npts` is the number of equally-spaced points in the time region. By default, `n` = 100, `tf` = 3T where `T` is the period of the force, and `npts` = 301. The parameters `n`, `tf`, and `npts` can be replaced by [ ] as a placeholder for default value.

`[y, t] = periodic(pLoad, n, tf, npts)` returns the steady-state displacement in a vector `y` and the times of computation in a vector `t`. With `y` and `t`, the steady-state displacement can be plotted by `plot(t, y)` or `plotvib(y)`, where `plot` is a standard MATLAB function, and `plotvib` is given in Window 3.12.

Note. Listed in Table 10.6.1 are five types of periodic loads. The load specification vector `pLoad` is of the form

```
pLoad = [load_ID T par1 par2 ...]
```

where `load_ID` is an integer from 0 to 4 identifying the load type, `T` is the period of the load, and `par1`, `par2`... are parameters that describe the pattern of the load. For `load_ID` = 1, 2, 3, and 4, exact Fourier coefficients are used in computation; see Section 10.6.3. For `load_ID` = 0, the numerical integration, Eq. (10.109), is used to determine Fourier coefficients.

EXAMPLE 6.1

A system of parameters $m = 1$, $\omega_n = 4$, and $\xi = 0.2$ is subject to a triangular periodic force of period $T = 4$; see Fig. 10.6.2. According to Table 10.6.1, the force is of Type P2. By

```
>> set1dof2(4,0.2)
>> pLoad = [2 4 3.0 0 2 4];
>> periodic(pLoad)
```

the steady-state displacement of the system is plotted in Fig. 10.6.3.

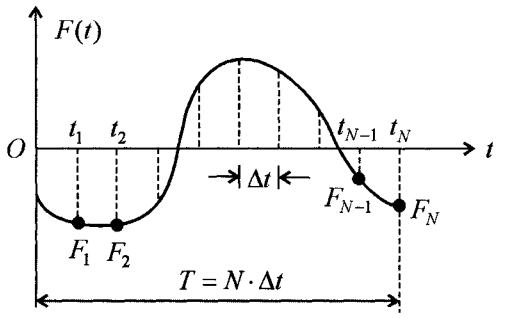
The Fourier coefficients of the force can also be obtained numerically according to Type P0 in Table 10.6.1. In this case, the steady-state displacement of the system can be obtained by the following commands:

```
>> n = 100;
>> f1=3/2*linspace(2/n, 2, n);
>> f2=3-3/2*(linspace(2+2/n,4,n)-2);
>> load_num = [0 4 f1 f2];
>> periodic(load_num)
```

Here, 200 time points in a period have been selected for computing the Fourier coefficients of the periodic force.

TABLE 10.6.1 Periodic Load $F(t)$ and Load Specification Vector pLoad

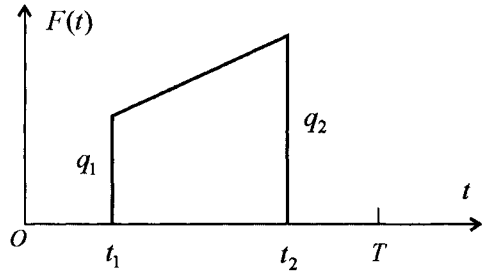
Load P0 Numerical data



pLoad = [0 T F_1 $F_2 \cdots F_N$]

Here $F_i = F(t_i)$ with $t_i = i \cdot \Delta t$
and $\Delta t = T/N$

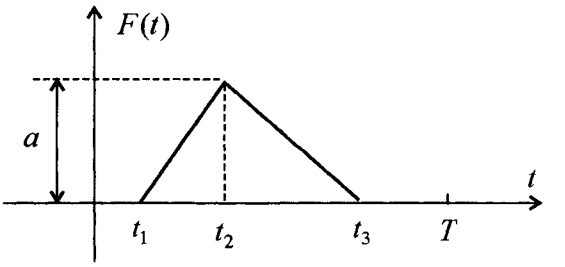
Load P1



$0 \le t_1 < t_2 \le T$; q_1 and q_2 arbitrary numbers

pLoad = [1 T t_1 q_1 t_2 q_2]

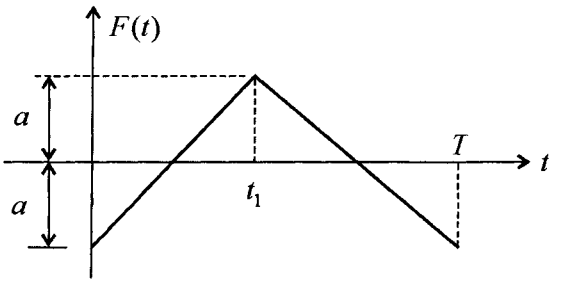
Load P2



$0 \le t_1 < t_2 < t_3 \le T$; a arbitrary number

pLoad = [2 T a t_1 t_2 t_3]

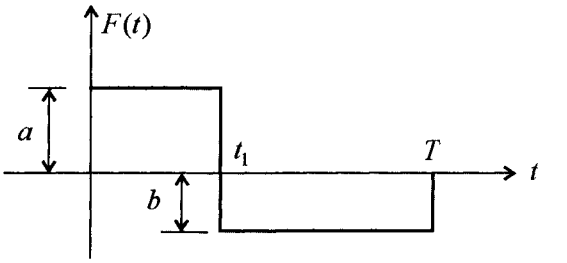
Load P3



$0 < t_1 < T$; a arbitrary number

pLoad = [3 T a t_1]

Load P4



$0 \le t_1 \le T$; a and b arbitrary numbers

pLoad = [4 T a b t_1]

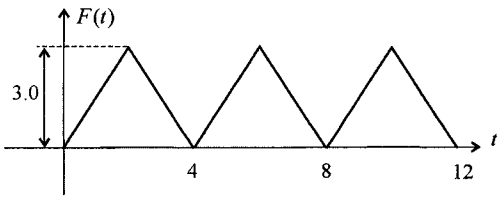


FIGURE 10.6.2

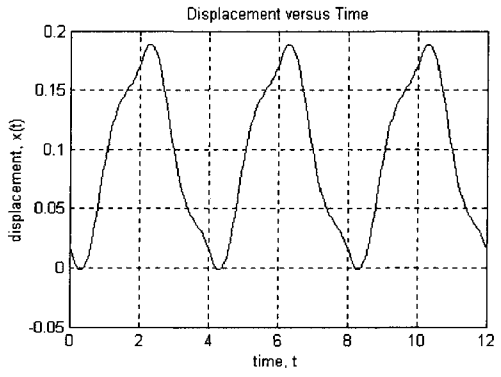


FIGURE 10.6.3

Let $x_a(t)$ be the analytical solution determined based on Type P2. Let $x_n(t)$ be the numerical solution obtained based on Type P0. The difference $x_a(t) - x_n(t)$ is plotted against time in Fig. 10.6.4 by

```
>> xa = periodic(pLoad);
>> [xn, t] = periodic(load_num);
>> plot(t, xa-xn)
>> xlabel('t')
>> ylabel('x_a(t) - x_n(t)')
```

which shows that the deviation of the numerical solution from the analytical solution is quite small.

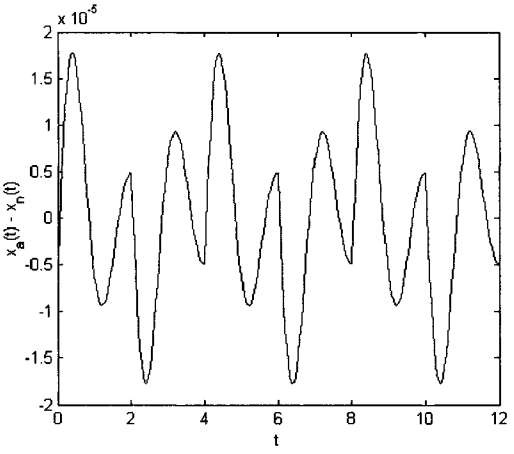


FIGURE 10.6.4

Given a periodic forcing function, its Fourier series, Eq. (10.100) can be obtained by function `fseries`; see Window 6.2.

Window 6.2. Function `fseries`

MATLAB Function: **`fseries`**

Purpose:
To obtain and plot Fourier series of a periodic forcing function listed in Table 10.6.1.

Synopsis:

```
fseries(pLoad)
fseries(pLoad, n, tf, npts)
[f, t, F_coeffs] = fseries(pLoad, n, tf, npts)
```

Description:
`fseries(pLoad)` obtains the Fourier coefficients of a periodic forcing function specified by vector `pLoad` according to Table 10.6.1.

`fseries(pLoad, n, tf, npts)` obtains coefficients $a_0 \quad a_1 \cdots a_n \quad b_1 \; b_2 \cdots b_n$ of the first 2n+1 terms of the Fourier series of a periodic forcing function and plots the series in the time region $0 \leq t \leq$ `tf`, where `npts` is the number of equally-spaced points in the time region selected for computation. By default, `n` = 100, `tf` = 3T where T is the period of the force, and `npts` = 301. The parameters `n`, `tf`, and `npts` can be replaced by [ ] as a placeholder for default value.

`[f, t, F_coeffs] = fseries(pLoad, n, tf, npts)` returns the values of the forcing function in a vector `f`, the times of computation in a vector `t`, and the Fourier coefficients of the forcing function in a vector `F_coeffs`.

If a periodic force can be decomposed into N periodic components that are listed in Table 10.6.1, the steady-state displacement can be obtained by superposition:

```
y = periodic(load1) + periodic(load2) + ··· + periodic(loadN)
```

where the vectors `load1, load2, ..., loadN` specify these N periodic components according to Table 10.6.1. Similarly, the Fourier coefficients of the periodic force can be obtained by

```
f = fseries(load1) + fseries(load2) + ··· + fseries(loadN)
```

The Fourier series so computed can be plotted by a function `plotpforce(f)` from the Toolbox.

The above discussion is summarized in Window 6.3.

Window 6.3. Simulation of Response to Periodic Loads

External Load: Periodic force

MATLAB Commands:

(i) To determine the Fourier series of a periodic force, use the commands

```
f = fseries(load1) + fseries(load2) + ··· + fseries(loadN)
plotpforce(f)
```

(ii) To determine the steady-state displacement, use the commands

```
setldof(m,c,k)
y = periodic(load1) + periodic(load2) + ··· + periodic(loadN)
plotvib(f)
```

Here `load1, load2, ..., loadN` are load specification vectors that are formed by Table 10.6.1.

EXAMPLE 6.2

A system with $m = 2$, $c = 2$, and $k = 4$ is subject to the periodic force shown in Fig. 10.6.5. The force can be decomposed into two periodic components of Type P1:

```
>> ld1 = [1 3 0 1.2 1 1.2];
>> ld2 = [1 3 1 1.2 2 0];
```

By

```
>> setldof(2, 2, 4)
>> y = periodic(ld1) + periodic(ld2);
>> plotvib(y)
```

the steady-state displacement of the system is plotted in Fig. 10.6.6. The Fourier series of the force is obtained by

```
>> f = fseries(ld1) + fseries(ld2);
>> plotpforce(f)
```

which yields the plot of the forcing function in Fig. 10.6.7, and the Fourier coefficients of each component as follows:

```
Load P1: pLoad = [2 T t1 q1 t2 q2]
         T = 3, t1 = 0, q1 = 1.2, t2 = 1, q2 = 1.2
  a0 = 0.8
    j         aj         bj
1.0000     0.3308     0.5730
2.0000    -0.1654     0.2865
3.0000    -0.0000          0
4.0000     0.0827     0.1432
5.0000    -0.0662     0.1146
6.0000    -0.0000          0
7.0000     0.0473     0.0819
8.0000    -0.0413     0.0716
9.0000    -0.0000          0
10.0000    0.0331     0.0573
```

```
Load P1: pLoad = [2 T t1 q1 t2 q2]
         T = 3, t1 = 1, q1 = 1.2, t2 = 2, q2 = 0

  a0 = 0.4

    j          aj          bj
1.0000     -0.3308      0.1249
2.0000      0.1654     -0.1745
3.0000      0.0000      0.1273
4.0000     -0.0827     -0.0280
5.0000      0.0662     -0.0508
6.0000      0.0000      0.0637
7.0000     -0.0473     -0.0208
8.0000      0.0413     -0.0288
9.0000      0.0000      0.0424
10.0000    -0.0331     -0.0159
```

The Fourier coefficients of the periodic force are the sums of the Fourier coefficients of the components.

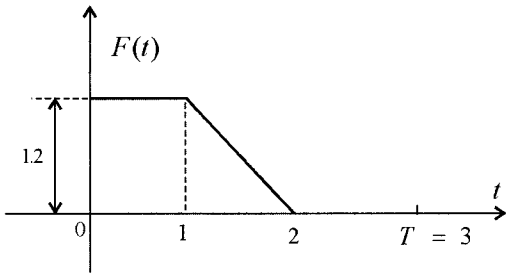


FIGURE 10.6.5

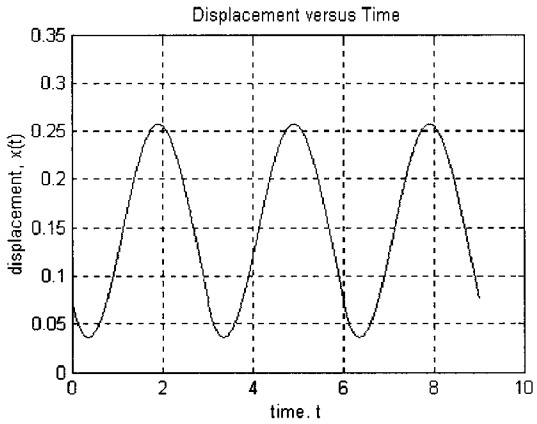


FIGURE 10.6.6

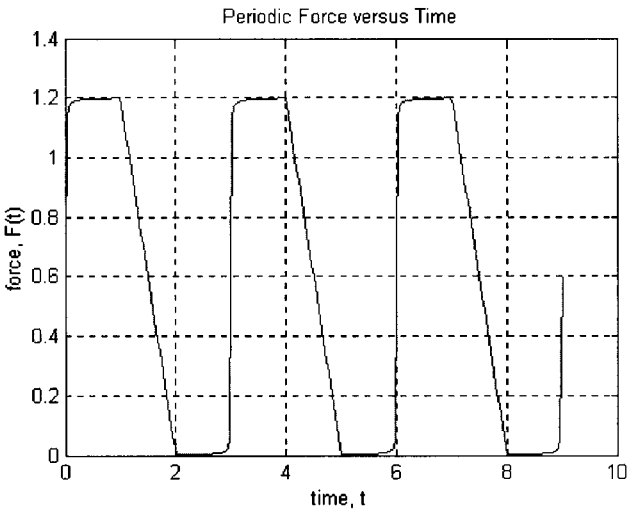


FIGURE 10.6.7

10.6.3 Fourier Coefficients of Certain Loads

As a reference, the Fourier coefficients of the loads P1 to P4 listed in Table 10.6.1 are given below.

Load P1:

$$a_0 = (q_1 + q_2)\frac{t_2 - t_1}{T}$$

$$\begin{aligned} a_k = {} & \frac{1}{k\pi}(q_1 - \beta_0 t_1)(\sin\omega_k t_2 - \sin\omega_k t_1) \\ & + \frac{\beta_0}{k\pi}\left[\frac{1}{\omega_k}(\cos\omega_k t_2 - \cos\omega_k t_1) + t_2\sin\omega_k t_2 - t_1\sin\omega_k t_1\right] \end{aligned} \tag{10.110}$$

$$\begin{aligned} b_k = {} & -\frac{1}{k\pi}(q_1 - \beta_0 t_1)(\cos\omega_k t_2 - \cos\omega_k t_1) \\ & + \frac{\beta_0}{k\pi}\left[\frac{1}{\omega_k}(\sin\omega_k t_2 - \sin\omega_k t_1) - t_2\cos\omega_k t_2 + t_1\cos\omega_k t_1\right] \end{aligned}$$

for $k = 1, 2, \ldots$, with

$$\beta_0 = \frac{q_2 - q_1}{t_2 - t_1} \quad \text{and} \quad \omega_k = \frac{2k\pi}{T}.$$

Load P2:

$$a_0 = \frac{1}{T}\left[\beta_1(t_2 - t_1)^2 + 2a(t_3 - t_2) - \beta_2(t_3 - t_2)^2\right]$$

$$a_k = -\frac{1}{k\pi}\left[\beta_1 t_1 (\sin\omega_k t_2 - \sin\omega_k t_1) + (a + \beta_2 t_2)(\sin\omega_k t_3 - \sin\omega_k t_2)\right]$$
$$+ \frac{\beta_1}{k\pi}\left[\frac{1}{\omega_k}(\cos\omega_k t_2 - \cos\omega_k t_1) + t_2 \sin\omega_k t_2 - t_1 \sin\omega_k t_1\right]$$
$$- \frac{\beta_2}{k\pi}\left[\frac{1}{\omega_k}(\cos\omega_k t_3 - \cos\omega_k t_2) + t_3 \sin\omega_k t_3 - t_2 \sin\omega_k t_2\right] \tag{10.111}$$

$$b_k = \frac{1}{k\pi}\left[\beta_1 t_1 (\cos\omega_k t_2 - \cos\omega_k t_1) - (a + \beta_2 t_2)(\cos\omega_k t_3 - \cos\omega_k t_2)\right]$$
$$+ \frac{\beta_1}{k\pi}\left[\frac{1}{\omega_k}(\sin\omega_k t_2 - \sin\omega_k t_1) - t_2 \cos\omega_k t_2 + t_1 \cos\omega_k t_1\right]$$
$$- \frac{\beta_2}{k\pi}\left[\frac{1}{\omega_k}(\sin\omega_k t_3 - \sin\omega_k t_2) - t_3 \cos\omega_k t_3 - t_2 \cos\omega_k t_2\right]$$

for $k = 1, 2, \ldots$, with

$$\beta_1 = \frac{a}{t_2 - t_1}, \quad \beta_2 = \frac{a}{t_3 - t_2} \quad \text{and} \quad \omega_k = \frac{2k\pi}{T}.$$

Load P3:

$$a_0 = \frac{1}{T}\left[2a(T - 2t_1) + (\beta_1 + \beta_2)t_1^2 - \beta_2 T^2 + 2\beta_2 t_1 (T - t_1)\right]$$
$$a_k = -\frac{2a}{k\pi}\sin\omega_k t_1 + \frac{1}{k\pi}(\beta_1 + \beta_2)\left[t_1 \sin\omega_k t_1 - \frac{1}{\omega_k}(1 - \cos\omega_k t_1)\right]$$
$$- \frac{\beta_2}{k\pi} t_1 \sin\omega_k t_1 \tag{10.112}$$
$$b_k = -\frac{2a}{k\pi}(1 - \cos\omega_k t_1) + \frac{1}{k\pi}(\beta_1 + \beta_2)\left[\frac{1}{\omega_k}\sin\omega_k t_1 - t_1 \cos\omega_k t_1\right]$$
$$+ \frac{\beta_2}{k\pi}\left[T - t_1 (1 - \cos\omega_k t_1)\right]$$

for $k = 1, 2, \ldots$, with

$$\beta_1 = \frac{2a}{t_1}, \quad \beta_2 = \frac{2a}{T - t_1} \quad \text{and} \quad \omega_k = \frac{2k\pi}{T}.$$

Load P4:

$$a_0 = 2\left[(a + b)\frac{t_1}{T} - b\right]$$
$$a_k = \frac{1}{k\pi}(a + b)\sin\omega_k t_1 \tag{10.113}$$
$$b_k = \frac{1}{k\pi}(a + b)(1 - \cos\omega_k t_1)$$

for $k = 1, 2, \ldots$, with $\omega_k = \frac{2k\pi}{T}$.

10.7 Nonlinear Vibration

This section is concerned with 1-DOF systems with nonlinear spring and damping elements. The motion of such a system, according to Fig. 10.7.1, is governed by

$$m\ddot{x}(t) = F(t) - F_d(t) - F_s(t) \tag{10.114}$$

along with the initial conditions $x(0) = x_0$ and $\dot{x}(0) = v_0$. Here, $F(t)$ is an external force, $F_d(t)$ is the damping force, and $F_s(t)$ is the spring force. In general, the damping and spring forces are nonlinear functions

$$F_d = F_d(x, \dot{x}), \quad F_s = F_s(x) \tag{10.115}$$

In Table 10.7.1 are several commonly used models of nonlinear damping and spring elements, where sgn(a) is the signum function, which is 1 for $a > 0$, -1 for $a < 0$, and 0 for $a = 0$.

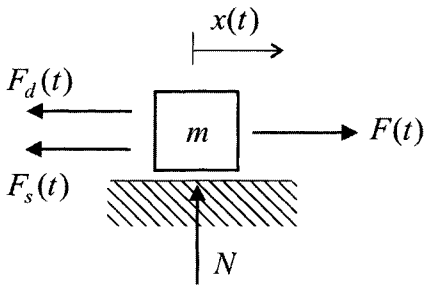


FIGURE 10.7.1

TABLE 10.7.1 Specification of Damping and Spring Elements

Damping Element	*Specification Vector*
Linear viscous damping: $F_d(t) = c\dot{x}$; c — damping coefficient	`Dmp_Spec` = c
Coulomb damping (dry friction): $F_d(t) = \mu N \text{sgn}(\dot{x})$ μ — kinetic friction coefficient, N — formal force	`Dmp_Spec` = [2 μ N]
Air damping (velocity-squared damping): $F_d(t) = c\dot{x}^2 \text{ sgn}(\dot{x})$ c — damping coefficient	`Dmp_Spec` = [3 c]
Spring Element	*Specification Vector*
Linear spring: $F_s(t) = kx$	`Spr_Spec` = k
Nonlinear spring: $F_s(t) = k_0 x + k_1 x^3$ $k_1 > 0$ for a hardening spring; $k_1 < 0$ for a softening spring	`Spr_Spec` = [2 k_0 k_1]
Restoring force of a pendulum: $F_s(t) = a_0 \sin(x)$ a_0 — either positive or negative	`Spc_Spec` = [3 a_0]

For a system with Coulomb damping, its response after some finite time t_f may be set at rest at one of many nonzero positions; namely, $x(t) = x_f \neq 0$ and $\dot{x}(t) = 0$ for $t \geq t_f$. This stiction or locking of vibration response at multiple equilibrium positions is due to the balance of the spring force $F_s(t)$, external force $F(t)$, and the *static* friction force F_c^{static} that is bounded by $-\mu_s N \leq F_c^{\text{static}} \leq \mu_s N$ where μ_s is a static friction coefficient. Impending vibration stiction can be predicted by the conditions: (i) $|F(t) - F_s(t)| < \mu_s N$; and (ii) $|\dot{x}(t)|$ is almost zero. For convenience in analysis, same static and kinetic friction coefficients are often assumed, i.e., $\mu = \mu_s$.

When a system of a nonlinear damping element is under a harmonic excitation

$$m\ddot{x} + F_d(x,\ \dot{x}) + kx = F_0 \sin \omega t \tag{10.116}$$

its steady-state response is often approximated by a linear system with an equivalent viscous damping coefficient c_{eq}; namely

$$m\ddot{x} + c_{eq}\dot{x} + kx = F_0 \sin \omega t \tag{10.117}$$

The equivalent damping coefficient is obtained by equating the energy dissipation per cycle; namely,

$$\Delta E = \int_0^{2\pi/\omega} F_d \dot{x}_{ss}\, dt = \int_0^{2\pi/\omega} (c_{eq}\dot{x}_{ss})\, \dot{x}_{ss}\, dt \tag{10.118}$$

where the steady-state response is assumed as

$$x_{ss}(t) = X \sin \omega t \tag{10.119}$$

The equivalent damping coefficient for Coulomb damping and air damping is given in Table 10.7.2, where X is the magnitude of the steady-state response and ω is the excitation frequency.

The motion of a nonlinear vibrating system can be determined numerically. Introduce the state variables

$$x_1(t) = x(t), \quad x_2(t) = \dot{x}(t) \tag{10.120}$$

by which, Eq. (10.114) is cast into an equivalent state-space form

$$\dot{z}(t) = \Phi(t,\ z(t)); \quad z(0) = z_0 \tag{10.121}$$

TABLE 10.7.2 Equivalent Viscous Damping Coefficient

Damping	F_d	c_{eq}
Linear viscous damping	$c\dot{x}$	c
Coulomb damping	$\mu N \text{sgn}(\dot{x})$	$\dfrac{4\mu N}{\pi \omega X}$
Air damping	$c\dot{x}^2 \text{sgn}(\dot{x})$	$\dfrac{8c\omega X}{3\pi}$

where

$$z(t) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad z_0 = \begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix}, \quad \Phi(t, z) = \begin{bmatrix} x_2 \\ \frac{1}{m}(F(t) - F_d(x_1, x_2) - F_s(x_1)) \end{bmatrix}.$$

The nonlinear state Eq. (10.121) can be solved by numerical integration algorithms, such as Runge-Kutta methods. A fixed-step Runge-Kutta method of order four is given below:

$$z_{k+1} = z_k + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4) \tag{10.122a}$$

where

$$\begin{aligned} f_1 &= \Phi(t_k, z_k) \\ f_2 &= \Phi\left(t_k + \frac{h}{2}, z_k + \frac{h}{2}f_1\right) \\ f_3 &= \Phi\left(t_k + \frac{h}{2}, z_k + \frac{h}{2}f_2\right) \\ f_4 &= \Phi(t_k + h, z_k + hf_3) \end{aligned} \tag{10.122b}$$

$z_k = z(t_k)$, $t_k = kh$, $k = 0, 1, 2, \ldots$, and h is the step size. In an advanced algorithm, the step size h is adaptively adjusted during the computation. The software MATLAB provides Runge-Kutta solvers like `ode45` with automatic step-size adjustment capability.

Solution by the Toolbox Numerical solution of Eq. (10.121) by the Toolbox takes three steps:

(i) Specification of a nonlinear system by function `setNL`;
(ii) Simulation of nonlinear vibration by function `NLvib`; and
(iii) Plotting of the computed response by function `plotvib`.

Functions `setNL` and `NLvib` are given in Windows 7.1 and 7.2.

Window 7.1. Function `set1NL`

MATLAB Function: **`set1NL`**

Purpose:
To specify the mass, damping, and spring elements of a 1-DOF nonlinear system.

Synopsis:

```
set1NL(m, Dmp_Spec, Spr_Spec)
```

Description:
`set1NL` specifies the mass `m` of the system, the damping element by a vector `Dmp_Spec`, and the spring element by a vector `Spr_Spec`. The formation of `Dmp_Spec` and `Spr_Spec` is given in Table 10.7.1.

Window 7.2. Function `NLvib`

MATLAB Function: **`NLvib`**

Purpose:
To obtain the response of a nonlinear 1-DOF system via numerical integration given in Eqs. (10.122).

Synopsis:

```
NLvib(x0, v0, Load_Spec)
NLvib(x0, v0)
[y, t] = NLvib(x0, v0, Load_Spec, tf, npts)
```

Description:
`NLvib(x0, v0, Load_Spec)` plots the total response of a nonlinear system subject to initial disturbances `x0` and `v0` and an external force specified by a vector `Load_Spec`. The formation of `Load_Spec` follows Table 10.3.1.

`NLvib(x0, v0)` plots the free response of a nonlinear system subject to initial disturbances `x0` and `v0`.

`[y, t] = NLvib(x0, v0, Load_Spec, tf, npts)` returns the computed system response in a two-row matrix `y` and the times of simulation in a vector `t`, where `tf` is the total time of simulation, and `npts` is the number of points in the time region $0 \leq t \leq$ `tf`. By default, `tf` is four times the natural period of the corresponding linear or linearized system, and `npts` = 501. If there is no external force, `Load_Spec` = 0. Also, `tf` can be replaced by [ ] as a placeholder for its default value.

With `y` and `t`, the time history of the nonlinear system can be plotted by the MATLAB function `plot` as follows:

```
plot(t, y(1,:))   for displacement
plot(t, y(2,:))   for velocity
```

The computed response can also be plotted by function `plotvib`; see Window 3.12 of Section 10.3.8.

With function `NLvib`, Eq. (10.121) is solved by the fixed-step Runge-Kutta numerical integration given in Eqs. (10.122). The step size of the algorithm is controlled by

$$h = \frac{\texttt{tf}}{\texttt{npts} - 1} \tag{10.123}$$

It is recommend that the user try several step sizes to compare the accuracy and convergence of numerical solutions.

EXAMPLE 7.1

A 1-DOF system of mass $m = 10$ kg and a linear spring $k = 100$ N/m is subject to Coulomb damping with kinetic friction coefficient $\mu = 0.1$ and normal force $N = 98.1$ N. Consider the free response of the system subject to two sets of initial

conditions: (1) $x_0 = 0.85\ m$ and $v_0 = 0$; (2) $x_0 = 0.70\ m$ and $v_0 = 0$. The free response of the system is plotted in Fig. 10.7.2 by

```
>> m = 10; Damp = [2 0.1 98.1]; k = 100;
>> setNL(m, Damp, k)
% Case 1
>> x0 = 0.85; v0 = 0;
>> [y1, t1] = NLvib(x0, v0, [ ], 5, 2001);
% Case 2
>> x0 = 0.7; v0 = 0;
>> [y2, t2] = NLvib(x0, v0, [ ], 5, 2001);
>> plot(t1, y1(1,:), t2, y2(1,:),':')
>> xlabel('time, t')
>> ylabel('x(t)')
>> title('Displacement versus Time')
>> grid
>> legend('x0 = 0.85', 'x0 = 0.7')
```

Due to the dry friction, the mass stops at different equilibrium positions, depending on the initial conditions.

Now assume that the system is under a triangular pulse shown in Fig. 10.7.3. With zero initial disturbances, the forced response of the system is plotted in Fig. 10.7.4 by

```
>> load = [5 0 0 5 100];
>> [y, t] = NLvib(0, 0, load, 15, 5001);
```

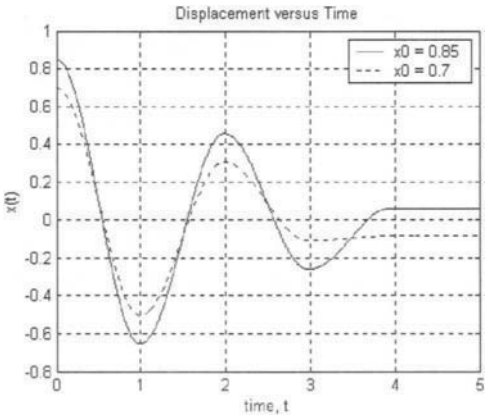


FIGURE 10.7.2

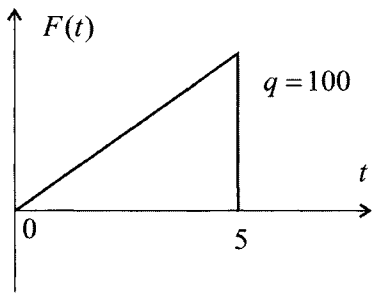


FIGURE 10.7.3

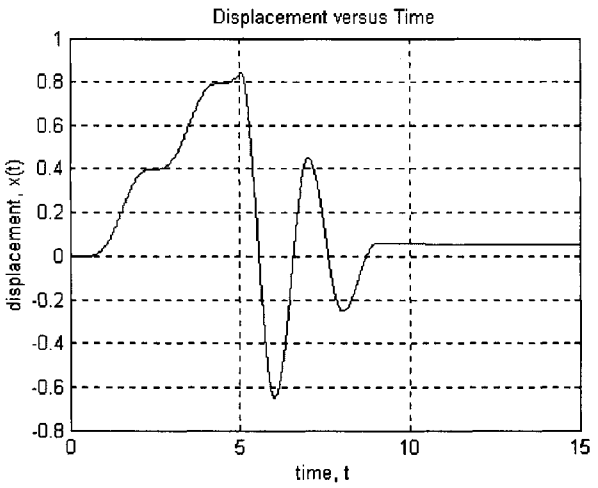


FIGURE 10.7.4

The displacement of the system is zero in the early stage as the external force is not large enough to overcome the static friction. At $t = \mu N/20 = 0.49$s, the system starts to move. After $t = 5$s, the system is in free vibration, and eventually settles at a nonzero position due to vibration stiction by the friction.

EXAMPLE 7.2

A compound pendulum in Fig. 10.7.5(a) is governed by the nonlinear equation

$$I_o\ddot{\theta} + c\dot{\theta} + mgr\sin\theta = 0 \tag{10.124}$$

where angle θ is measured from the straight down equilibrium position, I_o is the moment of inertia, and c is the coefficient of a rotational viscous damper (not shown in the figure). For "small" oscillations, a linearized model of the pendulum is often used:

$$I_o\ddot{\theta} + c\dot{\theta} + mgr\theta = 0 \tag{10.125}$$

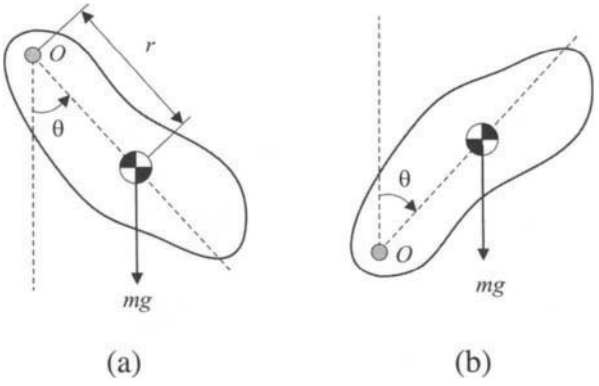


FIGURE 10.7.5 Compound pendulum.

Let the system parameters be

$$I_o = 12, \quad c = 5, \quad mgr = 40.$$

The free vibration of the pendulum is computed, to see how small the oscillations should be to use the linearized model. Consider zero initial velocity and different values of initial displacement. For $\theta(0) = \pi/3$ (30°),

```
>> x0 = pi/3; v0 = 0;
% Nonlinear model
>> setNL(12, 5, [3 40])
>> [y1, t] = NLvib(x0, v0, [ ], 20, 1001);
% Linear analysis
>> setNL(12, 5, 40)
>> y2 = NLvib(x0, v0, [ ], 20, 1001);
>> plot(t, y1(1,:)/pi*180, t, y2(1,:)/pi*180, ':')
>> legend('Nonlinear', 'Linear')
>> xlabel('t, sec'); ylabel('x(t), deg')
>> title('Displacement versus Time')
```

plots the free response in Fig. 10.7.6. Further simulation shows that the linearized model is in good agreement with the nonlinear model for $|(\theta)| < \pi/4$ (45°). Deviation occurs for larger values of initial displacement, as shown in Fig. 10.7.7, where the free response of the pendulum to $\theta(0) = \pi/3$ (60°) is plotted.

Now consider the inverted pendulum in Fig. 10.7.5(b), for which the nonlinear and linearized models are

$$I_o\ddot{\theta} + c\dot{\theta} - mgr\sin\theta = 0 \tag{10.126}$$

$$I_o\ddot{\theta} + c\dot{\theta} - mgr\theta = 0 \tag{10.127}$$

Here θ is measured from the straight-up equilibrium position. The free response of the nonlinear equation subject to the initial conditions

$$\theta(0) = 0, \quad \dot{\theta}(0) = 0.01$$

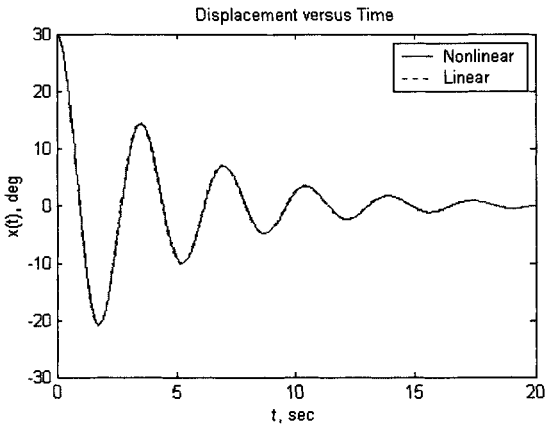


FIGURE 10.7.6 $\theta(0) = \pi/6$ (30°), $\dot{\theta}(0) = 0$.

is plotted in Fig. 10.7.8 by

```
>> setNL(12, 5, [3 -40])
>> [y, t] = NLvib(0, 0.01, [ ], 20, 1001);
>> plot(t, y(1,:)/pi*180)
>> xlabel('t, sec'); ylabel('x(t), deg')
```

Under the initial disturbance, the pendulum quickly jumps from the straight-up equilibrium position to the straight-down equilibrium position. This can be better seen from the phase plot in Fig. 10.7.9, which is obtained by

```
>> plot(y(1,:), y(2,:))
>> xlabel('x'); ylabel('dx/dt')
>> grid
```

Here y is the two-row matrix obtained by function `NLvib`.

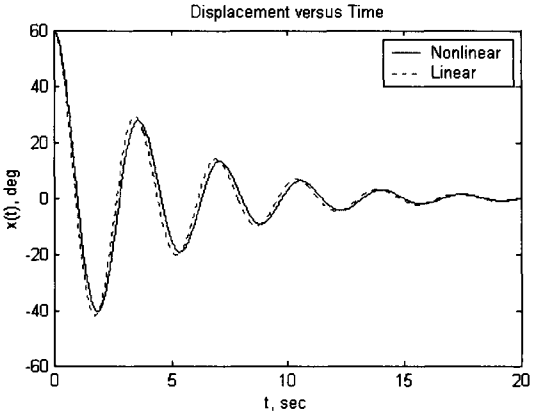


FIGURE 10.7.7 $\theta(0) = \pi/3$ (60°), $\dot{\theta}(0) = 0$.

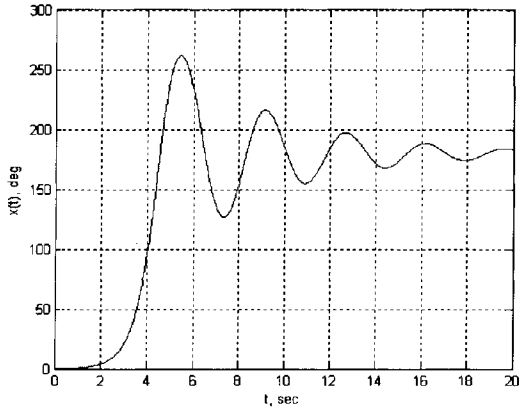


FIGURE 10.7.8 Free vibration of an inverted pendulum.

Finally, the free response of the nonlinear model is compared to that of the linearized model, both under the initial conditions $\theta(0) = 0, \dot{\theta}(0) = 0.01$.

```
>> setNL(12, 5, [3 -40]) % Nonlinear model
>> [y1, t] = NLvib(0, 0.01, [ ], 20, 1001);
>> setNL(12, 5, -40) % Linear model
>> y2 = NLvib(0, 0.01, [ ], 20, 1001);
>> semilogy(t, y1(1,:)/pi*180, t, y2(1,:)/pi*180, ':')
>> legend('Nonlinear', 'Linear')
>> xlabel('t, sec'); ylabel('x(t), deg')
```

yields Fig. 10.7.10. As expected, the linear vibration becomes unbounded because a negative spring coefficient exists in Eq. (10.127).

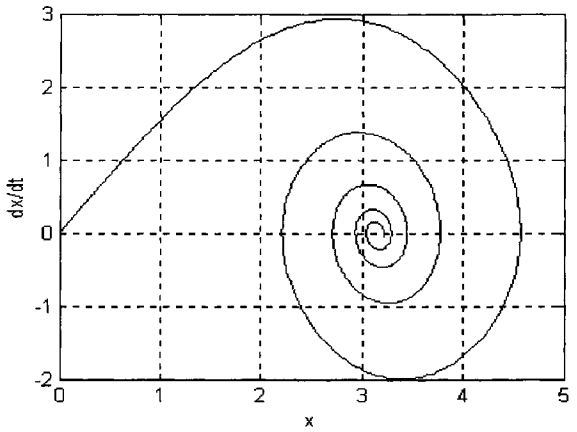


FIGURE 10.7.9 The phase plot of the inverted pendulum.

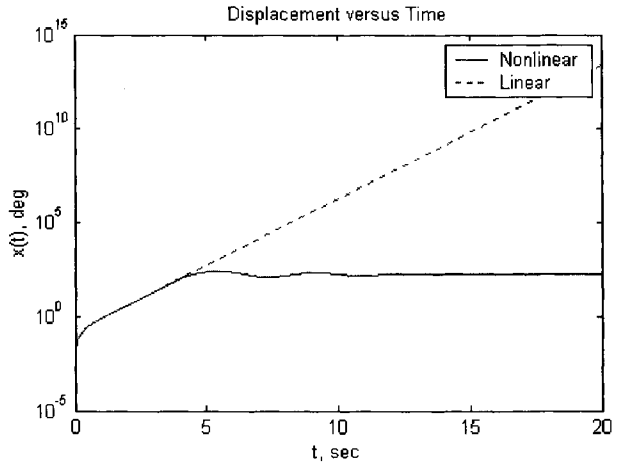


FIGURE 10.7.10

EXAMPLE 7.3

This example investigates the nonlinear effects of spring hardening and softening. Consider the following three systems:

System 1: linear spring $F_s = kx$, $k = 16$;
System 2: hardening spring $F_s = k_0 x + k_1 x^3$, $k_0 = 16$, $k_1 = 2$; and
System 3: softening spring $F_s = k_0 x + k_1 x^3$, $k_0 = 16$, $k_1 = -1$.

All the systems have mass $m = 1$ and linear viscous damping $c = 2$.

Under a step load $F(t) = 16$, the forced response of the systems is plotted in Fig. 10.7.11 by

```
>> m = 1; c = 2;
% system 1
>> setNL(m, c, 16)
>> [y1, t] = NLvib(0, 0, [1 16], 10, 1001);
% system 2
>> setNL(m, c, [2 16 2])
>> y2 = NLvib(0, 0, [1 16], 10, 1001);
% system 3
>> setNL(m, c, [2 16 -1])
>> y3 = NLvib(0, 0, [1 16], 10, 1001);
>> plot(t, y1(1,:), ':', t, y2(1,:), t, y3(1,:))
```

Compared to the linear spring, the softening spring leads to a larger steady-state response, and the hardening one leads to a smaller steady-state response.

Now, let the linear damping in Systems 2 and 3 be replaced by air damping, $F_c(t) = 2\dot{x}^2 \text{sgn}(\dot{x})$. The step response of the systems is plotted in Fig. 10.7.12, which is done by calling `setNL(1, [3 2], [2 16 2])` for System 2 and `setNL(1, [3 2], [2 16 -1])` for System 3. By comparing Figs. 10.7.11 and 10.7.12, one can see that air damping is less effective than viscous damping in settling transient response.

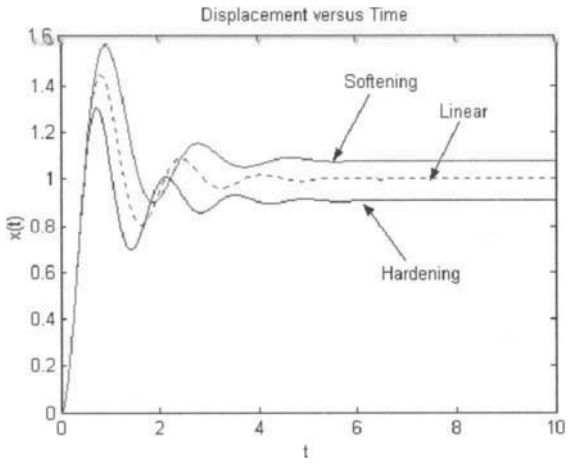


FIGURE 10.7.11

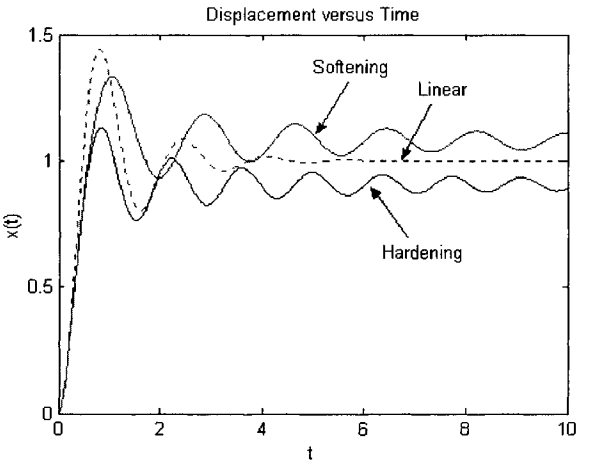
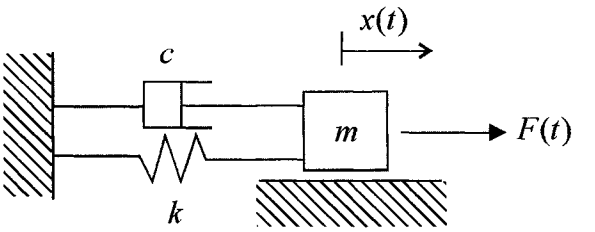


FIGURE 10.7.12

10.8 Quick Solution Guide

Description of 1-DOF Systems



Equation of motion

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t), \quad t \geq 0$$

Initial conditions

$$x(0) = x_0, \quad \dot{x}(0) = v_0$$

m — mass or inertia (kg),
c — damping coefficient (kg/s)
k — spring coefficient or stiffness (N/m)
$F(t)$ — external force
x_0 — initial displacement
v_0 — initial velocity.

Problems to Solve

This Toolbox provides functions for finding solutions of the following problems:

(a) Free vibration
(b) Forced vibration
(c) Frequency response
(d) Response to periodic excitation
(e) Nonlinear vibration

MATLAB Functions

	System Setup	
`setldof`	Input system parameters (m, c, k)	Window 2.1
`setldof2`	Input system parameters (ω_n, ξ, m)	Window 2.1
	Time Response	
`freevib`	Compute and display free response	Window 3.1
`forcedvib`	Compute and display forced response	Window 3.2
`stepcharact`	Compute maximum overshoot, rise time, and settling time of step response of an underdamped system	Section 10.3.3
`compltvib`	Plot total response due to both initial and external disturbances	Window 3.9
`energy`	Compute and display mechanical energy function	Window 3.11
`stepcharact`	Compute characteristic parameters of step response	Section 10.3.3
`plotvib`	Plot time response	Window 3.12
`getpts`	Get times of simulation	Window 3.13
	Analytical Vibration Solutions	
`forcedANL`	Obtain analytical expressions of forced response	Window 4.1
`plotANL`	Plot forced response by exact analytical solutions	Window 4.2
	Frequency Response	
`harmonic`	Plot frequency response to a harmonic excitation	Window 5.1
`plotfr`	Plot frequency response from computed data	Window 5.2
`normhar`	Plot normalized frequency response	Window 5.3
`transm`	Plot force or displacement transmissibility due to harmonic excitation	Window 5.4
`basevib`	Plot frequency response to a harmonic base excitation	Window 5.5
`unbalance`	Plot the frequency response of a rotating machine with an unbalanced mass	Window 5.6
`isolator`	Design the parameters of a vibration isolator	Window 5.7
	Response to Periodic Excitations	
`periodic`	Plot steady-state response to a periodic excitation	Window 6.1
`fseries`	Obtain the Fourier series for a periodic forcing function	Window 6.2
	Nonlinear Vibration	
`setNL`	Specify a 1-DOF nonlinear vibration system	Window 7.1
`NLvib`	Obtain numerical solution of nonlinear vibration	Window 7.2

(*continued*)

Utilities		
TBdemo	Show how the Toolbox works and what it can do	Section 10.1
TBinfo	Show the information of the Toolbox	Section 10.1
RunEx	Run all the numerical examples contained in this chapter	Section 10.1
systinfo	Display system information	Window 2.1

Solution by the Toolbox

To compute and display the response of a 1-DOF vibrating system, follow the three steps below.

Step 1. Input system parameters

```
>> set1dof(m,c,k)
```

where m is mass, c damping coefficient, and k spring coefficient

Step 2. Compute system response

For free response, type

```
>> y = freevib(x0,v0);
```

where x0 and v0 are initial displacement and velocity

For forced response, type

```
>> y = forcedvib(Load_Spec);
```

where Load_Spec specifies an external load

For the mathematical expressions of forced response, type

```
>> y = forcedANL(Load_Spec);
```

For frequency response to a harmonic excitation of magnitude f0, type

```
>> y = harmonic(f0);
```

For time-domain steady-state response to a periodic excitation, type

```
>> y = periodic(pLoad);
```

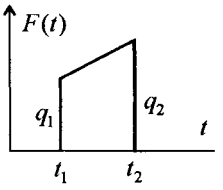
where pLoad specifies a periodic external load

Step 3. Plot computed response

Use `plotvib(y)` to plot time-domain response, and use `plotfr(y)` to plot frequency response, where y is a matrix of the computed response that is obtained in Step 2.

Note. For nonlinear vibration analysis, use functions `setNL` and `Nlvib`; see Section 10.7.

Specification of Vector Load_Spec for Functions ***forcedvib*** *and* ***compltvib***

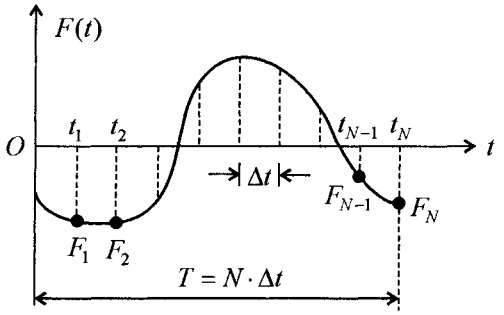
External Load	*F(t)*	Load_Spec
(L0) Impulse Force	$I_0\delta(t)$	[0 I_0 T_d]
	$I_0\delta(t-T_d)$	[0 I_0 T_d]
(L1) Step Force	F_0	[1 F_0]
	$F_0u(t-T_d)$	[1 F_0 T_d]
(L2) Ramp Force	$a_0\,t$	[2 a_0]
	$a_0\,(t-T_d)\,u(t-T_d)$	[2 a_0 T_d]
(L3) Exponential	$A\,e^{bt}$	[3 A b]
	$A\,e^{b(t-T_d)}\,u(t-T_d)$	[3 A b T_d]
(L4) Sinusoidal	$A\sin(\omega t+\phi_0)$	$A\sin(\omega t+\phi_0)$ [4 A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
	$A\sin(\omega(t-T_d)+\phi_0)\,u(t-T_d)$	[4 A ω ϕ_0 T_d] ω in rad/s, ϕ_0 in degrees
(L5) Pulse		[5 t_1 q_1 t_2 q_2]

Note 1. $\delta(t)$ is the delta function and $u(t)$ is the unite step function

Note 2. T_d is a nonnegative parameter representing time delay or time shift in a forcing function; see Section 10.3.5.

Specification of Vector pLoad for Function ***periodic***

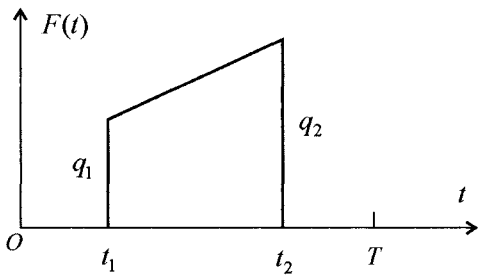
Load P0 Numerical data



pLoad = [0 T F_1 $F_2 \cdots F_N$]

Here $F_i = F(t_i)$ with $t_i = i \cdot \Delta t$ and $\Delta t = T/N$

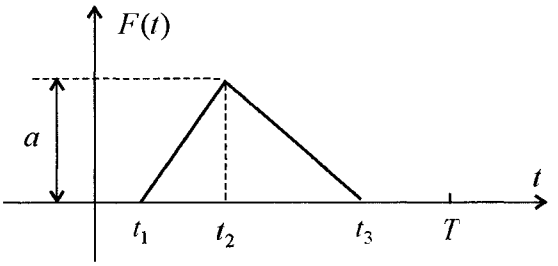
Load P1



$0 \le t_1 < t_2 \le T$; q_1 and q_2 arbitrary numbers

pLoad = [1 T t_1 q_1 t_2 q_2]

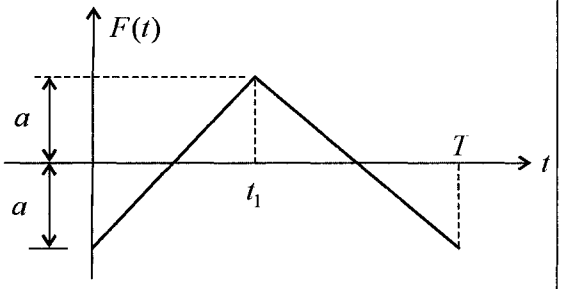
Load P2



$0 \le t_1 < t_2 < t_3 \le T$; a arbitrary number

pLoad = [2 T a t_1 t_2 t_3]

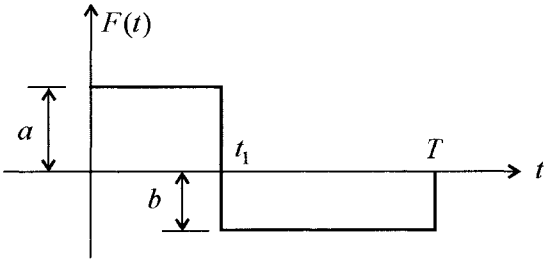
Load P3



$0 < t_1 < T$; a arbitrary number

pLoad = [3 T a t_1]

Load P4



$0 \le t_1 \le T$; a and b arbitrary numbers

pLoad = [4 T a b t_1]

10.9 References

1. Balachandran B, Magrab E B 2003 *Vibrations*, Brooks/Cole Publishing Co.: New York.
2. Den Hartog J P 1985 *Mechanical Vibrations*, Dover Publications: New York.
3. Inman D J 2000 *Engineering Vibrations*, 2nd ed., Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
4. Meirovitch L 1967 *Analytical Methods in Vibrations*, Macmillan: New York.
5. Rao S S 2003 *Mechanical Vibrations*, 4th ed., Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
6. Timoshenko S 1990 *Vibration Problems in Engineering*, John Wiley & Sons, 5th ed.
7. Dorf R C, Bishop R H 2000 *Modern Control Systems*, 9th ed., Prentice-Hall, Inc.: Upper Saddle River, New Jersey.

This Page Intentionally Left Blank

11 Vibration and Control of Multiple-Degree-of-Freedom Systems

Inside
- Getting Started
- System Description
- Dynamic Response
- Dynamic Vibration Absorption
- Transfer Function Formulation
- Feedback Control
- Quick Solution Guide
- References

11.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for vibration analysis and feedback control of linear mechanical systems with multiple degrees of freedom (M-DOF).

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on the computer
- The MATLAB Control System Toolbox needed to run certain functions (see Section 11.7)

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

(iii) Test the Toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

Quick Tutorial

To show how to use the Toolbox, consider a spring-mass-damper system in Fig. 11.1.1, with the values of the parameters given as

$$m_1 = m_2 = 1, \quad c_1 = 1, \quad c_2 = 2, \quad k_1 = 3, \quad k_2 = 4.$$

The equations of motion of the system are in the matrix form

$$\begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{bmatrix} 3 & -2 \\ -2 & 2 \end{bmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \begin{bmatrix} 7 & -4 \\ -4 & 4 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

To obtain the eigenvalues of the system, type the following in the MATLAB command window:

```
>> m=eye(2); c=[3 -2; -2 2]; k = [7 -4; -4 4];
>> setmdof(m, k, c)
```

where >> is the prompt in the MATLAB command window. This yields the eigenvalues of the system as follows

$$\lambda = -0.2255 \pm 1.0884i, \quad -2.2745 \pm 2.1307i$$

Let the external forces be

$$f_1(t) = 10, \quad f_2(t) = 0, \quad \text{for } t > 0$$

The forced vibration of the first mass is obtained by the following commands:

```
>> y = forcedRK([1 10], [1; 0]);
>> plotvib(y)
```

See Fig. 11.1.2 for the computed displacement $x_1(t)$.

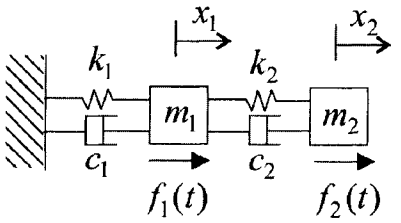


FIGURE 11.1.1

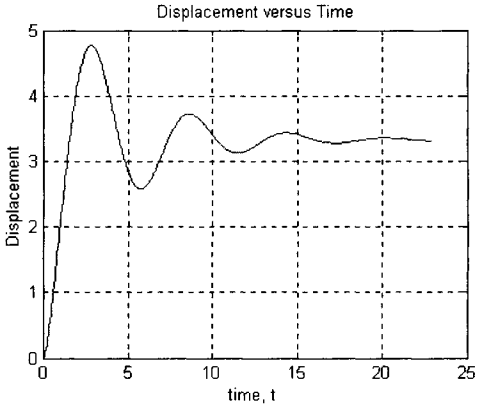


FIGURE 11.1.2

In the above vibration simulation, three functions from the Toolbox are called:

(a) Function **setmdof** that inputs system parameters and computes eigen solutions.
(b) Function **forcedRK** that computes the forced response of the system.
(c) Function **plotvib** that plots the computed displacement x_1 versus time.

These functions, along with others from the Toolbox, will be described in the subsequent sections.

For a quick solution, the reader may refer directly to the Quick Solution Guide (Section 11.7), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To better understand how the Toolbox works, the reader is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 11.8.

11.2 System Description

11.2.1 Introduction

Equations of Motion The vibration of a linear system of n degrees of freedom is governed by the second-order matrix differential equation

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \{F(t)\}, \quad t \geq 0 \tag{11.1}$$

subject to the initial conditions

$$\{x(0)\} = \{x_0\}, \quad \{\dot{x}(0)\} = \{v_0\}$$

where $\{x(t)\}$ is a vector of n displacements, $\{F(t)\}$ a force vector, $[M]$ an n-by-n inertia or mass matrix, $[C]$ an n-by-n damping matrix, and $[K]$ an n-by-n stiffness matrix. These parameter matrices are symmetric, namely, $[M] = [M]^T$, $[C] = [C]^T$, $[K] = [K]^T$. The mass matrix is positive definite, and the damping and stiffness matrices are positive semi-definite. When $\det[K] = 0$, there exist rigid-body modes in vibration.

EXAMPLE 2.1

For a 2-DOF spring-mass-damper system shown in Fig. 11.1.1, the equations of motion are

$$m_1\ddot{x}_1 + (c_1 + c_2)\dot{x}_1 - c_2\dot{x}_2 + (k_1 + k_2)x_1 - k_2x_2 = f_1(t)$$

$$m_2\ddot{x}_2 - c_2\dot{x}_1 + c_2\dot{x}_2 - k_2x_1 - k_2x_1 = f_2(t)$$

which can be cast into the matrix form (11.1), with

$$[M] = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}, \quad [C] = \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}, \quad [K] = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}.$$

Listed in Table 11.2.1 are three more examples of multiple-degree-of-freedom systems.

Eigenvalue Problem The eigenvalue problem of a multiple-degree-of-freedom (M-DOF) system is described by the homogeneous equation

$$\left(\lambda^2 [M] + \lambda [C] + [K]\right) \{u\} = 0 \tag{11.2}$$

where the unknown parameter λ is an eigenvalue, and the unknown vector $\{u\}$ is the associate eigenvector. The λ is a root of the characteristic equation

$$\det\left(\lambda^2 [M] + \lambda [C] + [K]\right) = 0 \tag{11.3}$$

which is a $2n$-th order polynomial of λ. So, a system of n degrees of freedom generally has $2n$ eigenvalues. The eigenpair $(\lambda, \{u\})$ describes a particular pattern of vibration by $\{x(t)\} = \{u\}e^{\lambda t}$, which is known as a *mode of vibration*. The eigensolutions (eigenvalues and eigenvectors) are useful for constructing the solution of Eq. (11.1).

System Classification by Damping Matrix M-DOF systems are often categorized by damping matrix in the following way:

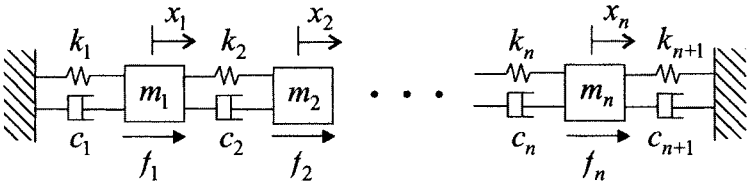
(i) Undamped systems with $[C] = 0$;
(ii) Proportionally damped systems with damping given in one of the following forms:
- Rayleigh damping $[C] = \alpha [M] + \beta [K]$, where α and β are constants;
- Modal damping; and
- General proportional damping that satisfies the condition $[C][M]^{-1}[K] = [K][M]^{-1}[C]$.

(iii) Nonproportionally damped systems, for which $[C][M]^{-1}[K] \neq [K][M]^{-1}[C]$.

The eigenvectors of an undamped or proportionally damped system are real; the eigenvectors of a nonproportionally damped system in general are complex.

TABLE 11.2.1 Examples of M-DOF Systems

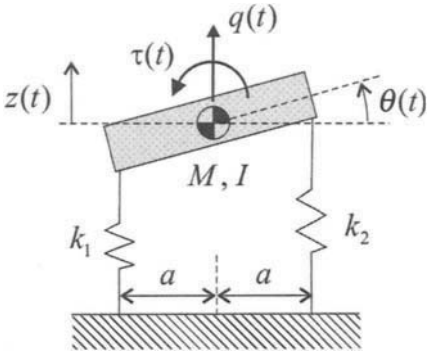
Spring-mass-damper systems



$$\begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & m_n \end{bmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \vdots \\ \ddot{x}_n \end{pmatrix} + \begin{bmatrix} c_1 + c_2 & -c_2 & 0 & \cdots & 0 \\ -c_2 & c_2 + c_3 & -c_3 & \cdots & 0 \\ 0 & -c_3 & c_3 + c_4 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & -c_n \\ 0 & \cdots & \cdots & -c_n & c_n + c_{n+1} \end{bmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{pmatrix}$$

$$+ \begin{bmatrix} k_1 + k_2 & -k_2 & 0 & \cdots & 0 \\ -k_2 & k_2 + k_3 & -k_3 & \cdots & 0 \\ 0 & -k_3 & k_3 + k_4 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & -k_n \\ 0 & \cdots & \cdots & -k_n & k_n + k_{n+1} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

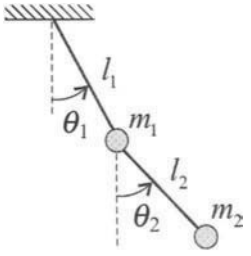
Elastically mounted rigid body



$$\begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \begin{pmatrix} \ddot{z} \\ \ddot{\theta} \end{pmatrix} + \begin{bmatrix} k_1 + k_2 & a(k_2 - k_1) \\ a(k_2 - k_1) & a^2(k_1 + k_2) \end{bmatrix} \begin{pmatrix} z \\ \theta \end{pmatrix} = \begin{pmatrix} q \\ \tau \end{pmatrix}$$

(small oscillations)

Double pendulum



$$\begin{bmatrix} (m_1 + m_2)l_1^2 & m_2 l_1 l_2 \\ m_2 l_1 l_2 & m_2 l_2^2 \end{bmatrix} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} + \begin{bmatrix} (m_1 + m_2)gl_1 & 0 \\ 0 & m_2 g l_2 \end{bmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

(small oscillations)

Solution Methods A fundamental problem in dynamic analysis of M-DOF systems is to solve Eq. (11.1) for $\{x(t)\}$ under external forces $\{F(t)\}$ and initial disturbances $\{x_0\}$ and $\{v_0\}$. Many analytical and numerical methods are available for the solution of Eq. (11.1). In Section 11.3, the following three solution techniques are presented:

- Modal analysis, which presents system response in a series of eigenvectors;
- Laplace transform method; and
- Numerical integration via a Runge-Kutta algorithm.

11.2.2 Modes of Vibration

In this section, modes of vibration are further discussed for systems with or without damping.

Undamped Systems For an undamped system ($[C] = 0$), Eq. (11.2) reduces to

$$\omega^2 [M]\{u\} = [K]\{u\} \tag{11.4}$$

where $\lambda^2 = -\omega^2$ has been used. The characteristic equation is

$$\det\left(-\omega^2 [M] + [K]\right) = 0 \tag{11.5}$$

which has n nonnegative roots or eigenvalues, $0 \leq \omega_1^2 \leq \omega_2^2 \leq \cdots \leq \omega_n^2$. The parameters $\omega_1, \omega_2, \ldots, \omega_n$ are called the *natural frequencies* of the system. The associate eigenvectors $\{u_1\}, \{u_2\}, \ldots, \{u_n\}$, which are always real, are called *mode shapes.*

For an undamped system with distinct eigenvalues, its eigenvectors are in the orthonormal relations

$$\{u_j\}^T [M]\{u_k\} = \delta_{jk}, \quad \{u_j\}^T [K]\{u_k\} = \delta_{jk}\omega_k^2 \tag{11.6}$$

where δ_{jk} is the Kronecker delta defined by

$$\delta_{jk} = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases}$$

Equations (11.6) can also be expressed as

$$[P]^T [M][P] = [I], \quad [P]^T [K][P] = \operatorname*{diag}_k \left\{\omega_k^2\right\} \tag{11.7}$$

where $[P]$ is the modal matrix formed by the eigenvectors

$$[P] = \begin{bmatrix} \{u_1\} & \{u_2\} & \cdots & \{u_n\} \end{bmatrix} \tag{11.8}$$

$[I]$ is an identity matrix, and $\operatorname*{diag}_k \left\{\omega_k^2\right\}$ is a diagonal matrix. For an undamped system with repeated eigenvalues, if all eigenvectors are linearly independent, the orthonormal relations (11.6) or (11.7) still hold.

Systems with Rayleigh Damping With the damping matrix of the form $[C] = \alpha [M] + \beta [K]$, Eq. (11.2) becomes

$$\left\{\lambda^2 [M] + \lambda (\alpha [M] + \beta [K]) + [K]\right\} \{u\} = 0 \tag{11.9}$$

The eigenvectors are the same as those of the undamped system, namely $\{u\} = \{u_k\}$, $k = 1, 2, \ldots, n$, with $\{u_k\}$ from Eq. (11.4). The characteristic equation corresponding to $\{u_k\}$ is

$$\lambda^2 + \lambda \left(\alpha + \beta\omega_k^2\right) + \omega_k^2 = 0 \tag{11.10}$$

which has the roots

$$\lambda_{\pm k} = -\frac{1}{2}\left(\alpha + \beta\omega_k^2\right) \pm i\sqrt{\omega_k^2 - \left(\alpha + \beta\omega_k^2\right)^2/4}, \quad i = \sqrt{-1} \tag{11.11}$$

Thus, for each eigenvector, there are two eigenvalues. The constants α and β are usually "small" such that $\lambda_{\pm k}$ are complex conjugate roots. If $\omega_k \neq 0$, Eq. (11.10) can be written as

$$\lambda^2 + 2\lambda\xi_k\omega_k + \omega_k^2 = 0 \tag{11.12}$$

where ξ_k is the damping ratio given by

$$\xi_k = \frac{1}{2}\left(\frac{\alpha}{\omega_k} + \beta\omega_k\right) \tag{11.13}$$

Systems with Modal Damping Modal damping is specified based on the assumption

$$[P]^T [C] [P] = \operatorname*{diag}_k \{2\xi_k\omega_k\} \tag{11.14}$$

where ξ_k is the damping ratio introduced to the kth mode of vibration. With the assumption, the eigenvectors of the damped system are the same as those of the corresponding undamped system, namely $\{u_k\}$, $k = 1, 2, \ldots, n$; and the characteristic equation for the kth eigenvector is

$$\lambda^2 + 2\xi_k\omega_k\lambda + \omega_k^2 = 0 \tag{11.15}$$

which has two roots

$$\lambda_{\pm k} = -\xi_k\omega_k \pm i\sqrt{1 - \xi_k^2} \cdot \omega_k, \quad i = \sqrt{-1} \tag{11.16}$$

So, for each eigenvector, there are two eigenvalues.

General Proportionally Damped Systems Under the condition $[C][M]^{-1}[K] = [K][M]^{-1}[C]$, it can be shown that (Reference 7)

$$[P]^T [C] [P] = \operatorname{diag} \{c_k\} \tag{11.17}$$

The eigenvectors of the damped system are the same as those of the corresponding undamped system. For the eigenvector $\{u_k\}$, the characteristic equation is

$$\lambda^2 + c_k\lambda + \omega_k^2 = 0 \tag{11.18}$$

TABLE 11.2.2 Eigensolutions of Damped Systems

Damping Type	*Eigenvalues*	*Eigenvectors*
No damping $[C] = 0$	n real and nonnegative eigenvalues $0 \le \omega_1^2 \le \omega_2^2 \le \cdots \le \omega_n^2$	n real orthogonal eigenvectors
Rayleigh damping $[C] = \alpha [M] + \beta [K]$	$2n$ complex eigenvalues $\lambda_{\pm k} = -\sigma_k \pm i\sqrt{\omega_k^2 - \sigma_k^2}$ $\sigma_k = \left(\alpha + \beta\omega_k^2\right)/2$	n real orthogonal eigenvectors
Modal damping $[P]^T [C] [P] = \underset{k}{\text{diag}} \{2\xi_k\omega_k\}$	$2n$ complex eigenvalues $\lambda_{\pm k} = -\xi_k\omega_k \pm i\sqrt{1 - \xi_k^2} \cdot \omega_k$	n real orthogonal eigenvectors
General proportional damping $[C][M]^{-1}[K] = [K][M]^{-1}[C]$	$2n$ complex eigenvalues $\lambda_{\pm k} = \frac{1}{2}\left(-c_k \pm i\sqrt{4\omega_k^2 - c_k^2}\right)$	n real orthogonal eigenvectors
Nonproportional damping $[C][M]^{-1}[K] \ne [K][M]^{-1}[C]$	$2n$ complex eigenvalues $\lambda_k,\ k = 1, 2, \ldots, 2n$	$2n$ complex eigenvectors that are nonorthogonal

which has two roots

$$\lambda_{\pm k} = \frac{1}{2}\left(-c_k \pm i\sqrt{4\omega_k^2 - c_k^2}\right), \quad i = \sqrt{-1} \tag{11.19}$$

The damping ratio for the kth mode is

$$\xi_k = \frac{c_k}{2\omega_k} \tag{11.20}$$

Nonproportionally Damped Systems A damped system, for which $[C][M]^{-1}[K] \ne [K][M]^{-1}[C]$, has $2n$ complex eigenvalues λ_k and $2n$ complex eigenvectors $\{v_k\}$, $k = 1, 2, \ldots, 2n$. Unlike those of an undamped system, the eigenvectors $\{v_k\}$ of a nonproportionally damped system do not enjoy the orthogonality relations (11.7). The damping ratio for a complex eigenvalue λ_j can be evaluated by

$$\xi_j = \frac{\hat{c}}{2\sqrt{\hat{m}\hat{k}}} \tag{11.21}$$

where $\hat{m} = \{\bar{v}_j\}^T [M] \{v_j\}$, $\hat{c} = \{\bar{v}_j\}^T [C] \{v_j\}$, and $\hat{k} = \{\bar{v}_j\}^T [K] \{v_j\}$, with $\{\bar{v}_j\}$ being the complex conjugate of $\{v_j\}$.

The eigensolutions of the above-mentioned systems are summarized in Table 11.2.2.

11.2.3 Solution by the Toolbox

An M-DOF system can be analyzed by the Toolbox of this chapter. Function `setmdof` from the Toolbox is always called first, to set up the system and to compute system eigensolutions; see Window 2.1.

Window 2.1. Function setmdof

MATLAB Function: **setmdof**

Purpose:
To input the parameter matrices of a system of n degrees of freedom ($n \geq 2$), and to compute the eigenvalues and eigenvectors of the system.

Synopsis:

```
setmdof(M, K, Dmp_Spec)
setmdof(M, K)
```

Description:
`setmdof(M, K, Dmp_Spec)` undertakes the following tasks:

(a) It inputs the inertia and stiffness matrices (`M` and `K`);
(b) It specifies damping by `Dmp_Spec` with the following options:

Case 1. No damping [C] = 0, `Dmp_Spec` = 0 or [ ]
Case 2. Rayleigh damping [C] = a*[M] + b*[K], `Dmp_Spec` = [1 a b]
Case 3. Modal damping, `Dmp_Spec` = [2 zeta(1) zeta(2) ... zeta(m)]
Here zeta(j) is the damping ratio of the jth mode, i.e., ξ_j. If m < n, the mth, (m+1)th, ... , and nth modes have the same damping ratio zeta(m). For instance, `Dmp_Spec` = [2 0.1 0.05] specifies a damping ratio of 0.1 for the first mode and a damping ratio of 0.05 for all other modes.
Case 4. General damping matrix [C], `Dmp_Spec` = [C]

(c) It computes the eigenvalues and eigenvectors of the system; and
(d) It prepares for further computation of time response or frequency response.

`setmdof(M, K)` carries on the above-mentioned tasks (a), (c), and (d) for an undamped system.

Function `setmdof` must be called before any other functions from the Toolbox can be used. However, for a system in different cases of initial and/or external excitations, `setmdof` only needs to be called once.

To output the eigensolutions computed by function `setmdof`, the function `geteig` can be used; see Window 2.2.

Window 2.2. Function geteig

MATLAB Function: **geteig**

Purpose:
To output the eigensolutions of an M-DOF system computed by function `setmdof`.

Window 2.2. Function `geteig` (*Continued*)

Synopsis:

```
[lmd, v, zeta] = geteig
```

Description:

`[lmd, v, zeta] = geteig` returns system eigenvalues in a vector `lmd`, eigenvectors in a matrix `zeta`, and damping ratios of all modes of vibration in a vector `zeta`.

The vibration of an M-DOF system in its kth mode is expressed by $\{x(t)\} = \text{Re}\left(\{u_k\}e^{\lambda_k t}\right)$, where λ_k and $\{u_k\}$ are an eigenpair of Eq. (11.2). The Toolbox has a function `animode` for animating modes of vibration; see Window 2.3.

Window 2.3. Function `animode`

MATLAB Function: **`animode`**

Purpose:

To play animated vibration of an M-DOF system in a particular mode, $\{x(t)\} = \text{Re}\left(\{u_k\}e^{\lambda_k t}\right)$.

Synopsis:

```
animode(mode_no)
F = animode(mode_no, tf, n_frames)
```

Description:

`animode(mode_no)` plays animated vibration in the mode specified by the mode number `mode_no`. For example, `animode(1)` animates the vibration in the first mode.

`F = animode(mode_no, tf, n_frames)` returns a set of frames of animated vibration of mode `mode_no` in a matrix `F`, which can be played later on by the MATLAB function `movie(F)`. Here, `tf` is total animation time, and `n_frames` is the number of frames to be animated. By default, `tf` = 8*T, with T being the period of the mode, and `n_frames` = 97.

To get the information about the system that has been set up by `setmdof`, function `systinfo` from the Toolbox can be called at any time. Also, the characteristic equation $\det(s^2M + sC + K) = 0$ can be obtained by function `chareq` from the Toolbox.

EXAMPLE 2.2

Consider a 3-DOF system with the inertia and stiffness matrices

$$[M] = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 30 \end{bmatrix}, \quad [K] = \begin{bmatrix} 10 & -10 & 0 \\ -10 & 30 & -20 \\ 0 & -20 & 60 \end{bmatrix}$$

Assume that the system is undamped. Typing the following in the MATLAB command window

```
>> M = diag([7 10 30]);
>> K = [10 -10 0; -10 30 -20; 0 -20 60];
>> setmdof(M, K)
```

yields the natural frequencies of the system

$$\omega_1 = 0.72251, \quad \omega_2 = 1.3266, \quad \omega_3 = 2.0363$$

and the mode shapes

$$\{u_1\} = \begin{pmatrix} 0.2723 \\ 0.1728 \\ 0.0779 \end{pmatrix}, \quad \{u_2\} = \begin{pmatrix} 0.2238 \\ -0.0519 \\ -0.1441 \end{pmatrix}, \quad \{u_3\} = \begin{pmatrix} 0.1365 \\ -0.2597 \\ 0.0807 \end{pmatrix}$$

which satisfy the orthonormal condition $\{u_j\}^T [M] \{u_k\} = \delta_{jk}$. Also,

```
>> chareq
```

yields the following information about the characteristic equation $\det(s^2M + K) = 0$:

```
Characteristic equation (order n = 6):
--------------------------------------
 det(s^*M + s*C + K) = s^6 + a5 s^5 + ... + a1 s + a0 = 0
 with a5 = 0, a4 = 6.4286, a3 = 0, a2 = 10.381, a1 = 0, a0 = 3.8095
```

If Rayleigh damping $[C] = 0.1 \cdot [M] + 0.2 \cdot [K]$ is added, the command

```
>> setmdof(M, K, [1 0.1 0.2])
```

produces the system eigenvalues

$$\lambda_{\pm 1} = -0.1022 \pm i0.7152; \quad \lambda_{\pm 2} = -0.2260 \pm i1.3072; \quad \lambda_{\pm 3} = -0.4647 \pm i1.9826.$$

The eigenvector $\{u_k\}$ associate with $\lambda_{\pm k}$ is the same as that of the corresponding undamped system. To display the information on the system eigensolutions, type

```
>> systinfo
```

which gives the following results:

```
Eigenvalues of a proportionally damped system ((lambda^2*M +
lambda*C + K)*V = 0):
-------------------------------------------------------------
 omega = natural frequencies in rad/sec, omega^2*M*U = K*U
 zeta = damping ratio
 wd = damped frequency = imag(lambda)
 lambda_1, lambda_2 = eigenvalues
```

```
  omega      zeta       wd          lambda_1          lambda_2
  0.7225     0.1415     0.7152    -0.1022 + 0.7152i  -0.1022 - 0.7152i
  1.3266     0.1704     1.3072    -0.2260 + 1.3072i  -0.2260 - 1.3072i
  2.0363     0.2282     1.9826    -0.4647 + 1.9826i  -0.4647 - 1.9826i

 Eigenvectors [V1 V2 ...]:
  0.2723     0.2238     0.1365
  0.1728    -0.0519    -0.2597
  0.0779    -0.1441     0.0807

 Here eigenvectors Vj are the same as those of the undamped system
 (M,K)
```

Now, assume that modal damping is introduced as follows: $\xi_1 = 0.1, \quad \xi_2 = \xi_3 = 0.05$. The command

```
>> setmdof(M, K, [2 0.1 0.05])
```

yields the eigenvalues of the system

$\lambda_{\pm 1} = -0.0723 \pm i0.7189; \quad \lambda_{\pm 2} = -0.0663 \pm i1.3249; \quad \lambda_{\pm 3} = -0.1018 \pm i2.0338.$

The eigenvectors are the same as those of the corresponding undamped system.

Finally, consider the damping matrix

$$[C] = \begin{bmatrix} 4 & -2 & 0 \\ -2 & 6 & -4 \\ 0 & -4 & 10 \end{bmatrix}$$

which makes the system nonproportionally damped because $[C][M]^{-1}[K] \neq [K][M]^{-1}[C]$. By

```
>> C = [4 -2 0; -2 6 -4; 0 -4 10];
>> setmdof(M, K, C)
```

the eigensolutions are obtained as follows:

$$\lambda_1 = -0.1218 + i0.7190, \quad \{v_1\} = \begin{pmatrix} 0.6644 \\ 0.4160 - i0.0511 \\ 0.1842 - i0.0420 \end{pmatrix}$$

$$\lambda_2 = \bar{\lambda}_1, \quad \{v_2\} = \{\bar{v}_1\}$$

$$\lambda_3 = -0.2044 + i1.3011, \quad \{v_3\} = \begin{pmatrix} -0.0773 + i0.4923 \\ 0.1164 - i0.0784 \\ 0.1150 - i0.2906 \end{pmatrix}$$

$$\lambda_4 = \bar{\lambda}_3, \quad \{v_4\} = \{\bar{v}_3\}$$

$$\lambda_5 = -0.4262 + i1.9869, \quad \{v_5\} = \begin{pmatrix} 0.0029 - i0.2024 \\ -0.0787 + i0.3668 \\ 0.0337 - i0.1100 \end{pmatrix}$$

$$\lambda_6 = \bar{\lambda}_5, \quad \{v_6\} = \{\bar{v}_5\}$$

The eigenvectors are complex, and are not orthogonal in the sense of Eq. (11.6).

EXAMPLE 2.3

Consider, the 3-DOF system with nonproportional damping in Example 2.2. The commands

```
>> M = diag([7 10 30]); K = [10 -10 0; -10 30 -20; 0 -20 60];
>> C = [4 -2 0; -2 6 -4; 0 -4 10];
>> setmdof(M, K, C)
>> animode(3)
```

play the animated vibration in the mode of the third eigenvalue $\lambda_3 = -0.2044 + i1.3011$ and associate eigenvector. Figure 11.2.1 shows some frames of the animation.

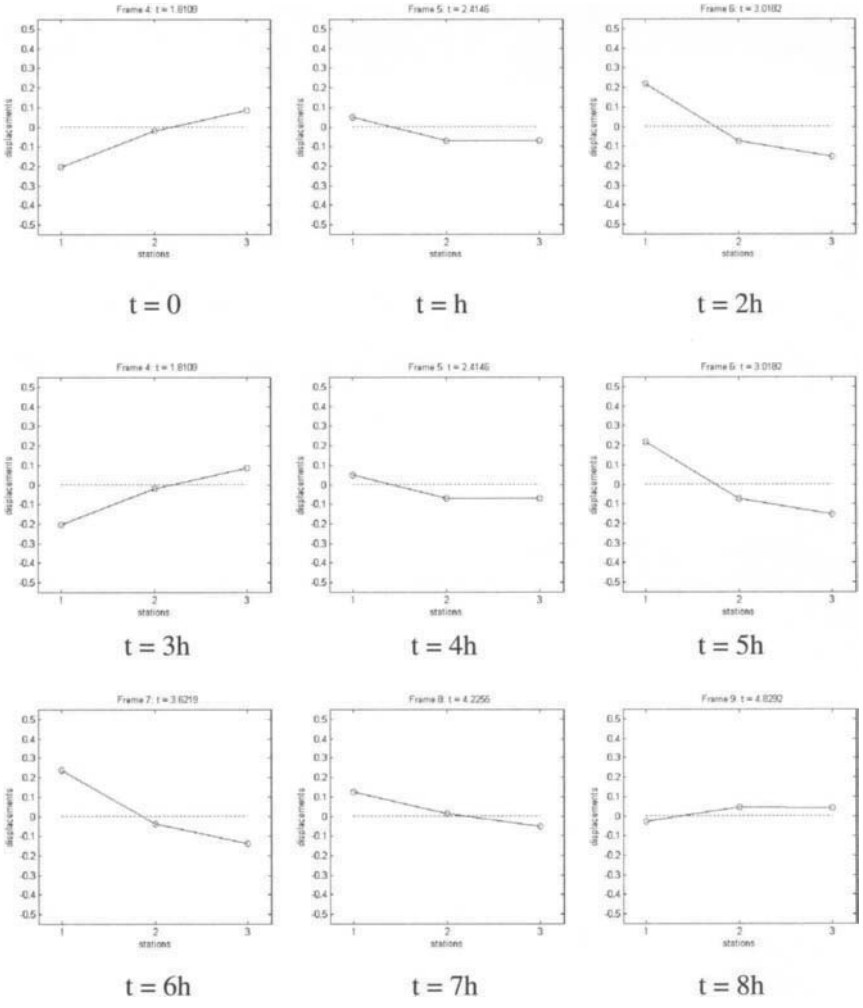


FIGURE 11.2.1 Animation of mode $(\lambda_3, \{v_3\})$, $h = T/8$, $T = 2\pi/\text{imag}(\lambda_3) = 4.8292$.

11.3 Dynamic Response

Consider a linear system of multiple-degrees-of-freedom (M-DOF)

$$\begin{aligned}[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} &= \{B_f\} f(t), \quad t \geq 0 \\ \{x(0)\} = \{x_0\}, \quad \{\dot{x}(0)\} &= \{v_0\}\end{aligned} \tag{11.22}$$

where $f(t)$ is a forcing function and $\{B_f\}$ is a force distribution vector. In this section, three commonly used solution methods for the above equation are presented: modal analysis, Laplace transform method, and Runge-Kutta algorithm. The first two methods are analytical techniques; the last one is a numerical integration method. In addition, the steady-state response of an M-DOF system to a harmonic excitation is determined at the end of this section.

11.3.1 Modal Analysis

In modal analysis, the displacements of an M-DOF system are expressed as a series of system eigenvectors with unknown modal coordinates to be determined. Based on certain orthogonality relations among the eigenvectors, the original matrix equation of motion, Eq. (11.22), is decoupled to a number of scalar differential equations about the unknown coordinates. Determination of these equations eventually yields the dynamic response of the system. This solution technique is also called *modal expansion* or *eigenfunction expansion.*

Undamped Systems The response of an undamped system ($[C] = 0$) is expressed by

$$\{x(t)\} = \sum_{k=1}^{n} \{u_k\} q_k(t) = [P]\{q(t)\} \tag{11.23}$$

where $[P]$ is the modal matrix given in Eq. (11.8), and

$$\{q(t)\} = \begin{pmatrix} q_1(t) & q_2(t) & \cdots & q_n(t) \end{pmatrix}^T \tag{11.24}$$

with $q_j(t)$ being the unknown modal coordinates. Substitution of the modal expansion (11.23) into Eq. (11.22) with $[C] = 0$ and use of the orthogonality relations (11.7) yields

$$\begin{aligned}\{\ddot{q}(t)\} + \operatorname*{diag}_k \{\omega_k^2\}\{q(t)\} &= [P]^T \{B_f\} f(t) \equiv \{r\} f(t), \quad t \geq 0 \\ \{q(0)\} = [P]^T [M]\{x_0\} \equiv \{a_0\}, \quad \{\dot{q}(0)\} &= [P]^T [M]\{v_0\} \equiv \{b_0\}\end{aligned} \tag{11.25}$$

which consists of n decoupled differential equations about the modal coordinates

$$\begin{aligned}\ddot{q}_k(t) + \omega_k^2 q_k(t) &= r_k f(t) \\ q_k(0) = a_{0,k}, \quad \dot{q}_k(0) &= b_{0,k}\end{aligned} \tag{11.26}$$

$k = 1, 2, \ldots, n$. Here r_k, $a_{0,k}$, and $b_{0,k}$ are from the vectors

$$\{r\} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix}, \quad \{a_0\} = \begin{pmatrix} a_{0,1} \\ a_{0,2} \\ \vdots \\ a_{0,n} \end{pmatrix}, \quad \{b_0\} = \begin{pmatrix} b_{0,1} \\ b_{0,2} \\ \vdots \\ b_{0,n} \end{pmatrix} \tag{11.27}$$

The solution of Eq. (11.26) is given by

$$q_k(t) = a_{0,k}\dot{g}_k(t) + b_{0,k}g_k(t) + r_k \int_0^t g_k(t-\tau)f(\tau)\,d\tau \tag{11.28}$$

where the influence function

$$g_k(t) = \begin{cases} \dfrac{1}{\omega_k}\sin\omega_k t & \text{for } \omega_k \neq 0 \\ t & \text{for } \omega_k = 0 \end{cases} \tag{11.29}$$

Once the modal coordinates $q_k(t)$ are known, the response of the system can be estimated by the modal expansion (11.23).

Systems with Proportional Damping For a system with proportional damping ($[C][M]^{-1}[K] = [K][M]^{-1}[C]$), the use of the modal expansion (11.23) and the orthogonality relations (11.7) yields

$$\begin{aligned} &\ddot{q}_k(t) + c_k\dot{q}_k(t) + \omega_k^2 q_k(t) = r_k f(t) \\ &q_k(0) = a_{0,k}, \quad \dot{q}_k(0) = b_{0,k} \end{aligned} \tag{11.30}$$

where r_k, $a_{0,k}$, and $b_{0,k}$ are defined in Eq. (11.27). The damping coefficient c_k is given as follows

(i) For Rayleigh damping ($[C] = \alpha[M] + \beta[K]$), $c_k = \alpha + \beta\omega_k^2$;
(ii) For modal damping, $c_k = 2\xi_k\omega_k$; and
(iii) For general proportional damping, $c_k = \{u_k\}^T[C]\{u_k\}$, where $\{u_k\}$ is the kth eigenvector of the corresponding undamped system.

The solution of Eq. (11.30) is given by

$$q_k(t) = g_k(t)\left(c_k a_{0,k} + b_{0,k}\right) + \dot{g}_k(t)a_{0,k} + r_k \int_0^t g_k(t-\tau)f(\tau)d\tau \tag{11.31}$$

The influence function $g_k(t)$ in the above equation is given in the following two cases:

(i) $\omega_k \neq 0$ (flexible mode)

$$g_k(t) = \begin{cases} \dfrac{1}{\omega_{d,k}}e^{-\sigma_k t}\sin\omega_{d,k}t & \text{for } 0 \leq \xi_k < 1 \\ t\,e^{-\omega_k t} & \text{for } \xi_k = 1 \\ \dfrac{1}{\beta_k}e^{-\sigma_k t}\sinh\beta_k t & \text{for } \xi_k > 1 \end{cases} \tag{11.32}$$

where the damping ratio ξ_k is defined by $c_k = 2\xi_k\omega_k$, $\sigma_k = \xi_k\omega_k$, $\omega_{d,k} = \sqrt{1-\xi_k^2}\,\omega_k$, and $\beta_k = \sqrt{\xi_k^2 - 1}\,\omega_k$. Here, $0 \leq \xi_k < 1$ when the kth mode is undamped or underdamped; $\xi_k = 1$ when the kth mode is critically damped; $\xi_k > 1$ when the kth mode is overdamped. (Refer to Section 10.2 for damping status specification.)

(ii) $\omega_k = 0$ (rigid-body mode)

$$g_k(t) = \begin{cases} \dfrac{1}{c_k}\left(1 - e^{-c_k t}\right) & \text{for } c_k \neq 0 \\ t & \text{for } c_k = 0 \end{cases} \tag{11.33}$$

Also, refer to Chapter 10 for solution of Eq. (11.30) with various forcing functions.

Systems with Nonproportional Damping For a nonproportionally damped system, its eigenvectors do not enjoy the orthogonality relations (11.7); the modal expansion (11.23) cannot decouple Eq. (11.22). To find a usable modal expansion, the original equation (11.22) of motion is cast into a state form

$$\begin{aligned} \{\dot{z}(t)\} &= [A]\{z(t)\} + [B]f(t) \\ \{z(0)\} &= \{z_0\} \end{aligned} \tag{11.34}$$

where

$$\begin{aligned} \{z\} &= \begin{Bmatrix} \{x\} \\ \{\dot{x}\} \end{Bmatrix}, \quad \{z_0\} = \begin{Bmatrix} \{x_0\} \\ \{v_0\} \end{Bmatrix} \\ [A] &= \begin{bmatrix} 0 & [I] \\ -[M]^{-1}[K] & -[M]^{-1}[C] \end{bmatrix}, \quad [B] = \begin{bmatrix} \{0\} \\ [M]^{-1}\{B_f\} \end{bmatrix} \end{aligned} \tag{11.35}$$

$[I]$ is an identity matrix, and $\{0\}$ is a zero vector. The eigenvalue problem in the state space is

$$[A]\{\phi_k\} = \lambda_k \{\phi_k\} \tag{11.36}$$

Define an adjoint eigenvalue problem by

$$[A]^T \{\psi_k\} = \lambda_k \{\psi_k\} \tag{11.37}$$

Both Eqs. (11.36) and (11.37) have the same eigenvalues λ_k as those of the original problem described by Eq. (11.2). Also, the state eigenvectors can be expressed as

$$\{\phi_k\} = \begin{Bmatrix} \{v_k\} \\ \lambda_k \{v_k\} \end{Bmatrix}, \quad \{\psi_k\} = \begin{Bmatrix} (\lambda_k[M] + [C])\{v_k\} \\ \lambda_k[M]\{v_k\} \end{Bmatrix}, \quad k = 1, 2, \ldots, 2n \tag{11.38}$$

The eigenvectors $\{\phi_k\}$ and $\{\psi_k\}$ are in the bi-orthogonality relations

$$\{\psi_j\}^T \{\phi_k\} = \delta_{jk}, \quad \{\psi_j\}^T [A]\{\phi_k\} = \lambda_k \delta_{jk} \tag{11.39}$$

in which the eigenvectors have been scaled according to the normalization condition

$$\{\psi_k\}^T \{\phi_k\} = \{v_k\}^T [C]\{v_k\} + 2\lambda_k \{v_k\}^T [M]\{v_k\} = 1 \tag{11.40}$$

Assume that the response of the system is expressed as

$$z(t) = \sum_{k=1}^{2n} \{\phi_k\} q_k(t) \tag{11.41}$$

Substitute Eq. (11.41) into Eq. (11.34) and apply the bi-orthogonality relations (11.39), to obtain

$$\begin{aligned} \dot{q}_k(t) &= \lambda_k q_k(t) + \{\psi_k\}^T [B] f(t) = \lambda_k q_k(t) + r_k f(t) \\ q_k(0) &= \{\psi_k\}^T \{z_0\} \end{aligned} \tag{11.42}$$

The solution of (11.42) is given by

$$q_k(t) = e^{\lambda_k t} q_k(0) + r_k \int_0^t e^{\lambda_k (t-\tau)} f(\tau)\, d\tau \tag{11.43}$$

Solution by the Toolbox This Toolbox has functions `freeMD` and `forcedMD` for modal analysis of undamped and proportionally damped systems; see Windows 3.1 and 3.2. (For solution of nonproportionally damped systems, refer to Section 11.3.2.) To be able to run function `forcedMD`, the MATLAB Control System Toolbox is needed.

Window 3.1. Function freeMD

MATLAB Function: **freeMD**

Purpose:
To obtain the mathematical expressions of free response of an undamped or proportionally damped M-DOF system

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = 0, \quad t \geq 0$$
$$\{x(0)\} = \{x_0\}, \quad \{\dot{x}(0)\} = \{v_0\}$$

by modal expansion (11.23).

Synopsis:

```
freeMD(x0, v0)
[y, q] = freeMD(x0, v0)
```

Description:
`freeMD(x0, v0)` displays the mathematical expressions of the system free response subject to initial disturbances, where `x0` is a vector of initial displacements and `v0` a vector of initial velocities.

`[y, q] = freeMD(x0, v0)` returns the coefficients of the mathematical expressions of the free response $\{x(t)\}$ in a three-dimensional matrix `y`, and the coefficients of the mathematical expressions of the modal coordinates $\{q(t)\}$ in a three-dimensional matrix `q`. The meaning of the entries of `y` and `q` is explained in Table 11.3.1. The matrices `y` and `q` can be used to compute the free response by function `respCV` (see Window 3.5).

In executing `freeMD`, decoupled equations about modal coordinates like Eq. (11.26) are displayed in the MATLAB command window as a midstep in modal analysis.

Window 3.2. Function forcedMD

MATLAB Function: **forcedMD**

Purpose:
To obtain the mathematical expressions of forced response of an undamped or proportionally damped M-DOF system

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \{B_f\} f(t), \quad t \geq 0$$
$$\{x(0)\} = 0, \quad \{\dot{x}(0)\} = 0$$

by modal expansion (11.23). Here zero initial conditions are imposed.

Synopsis:

```
forcedMD(Load_Spec, Bf)
[y, q] = forcedMD(Load_Spec, Bf)
```

Description:
`forcedMD(Load_Spec, Bf)` displays the mathematical expressions of force response of the system subject to an external force $f(t)$ specified by a vector or matrix `Load_Spec`, where `Bf` is the load distribution vector. Specification of `Load_Spec` is given in Table 11.3.2. By default `Bf` $= (1 \quad 0 \quad 0 \quad \cdots \quad 0)^T$.

`[y, q] = forcedMD(Load_Spec, Bf)` returns the coefficients of the mathematical expressions of the forced response in a three-dimensional matrix `y` and the coefficients of the mathematical expressions of the modal coordinates in a three-dimensional matrix `q`. The meaning of the entries of `y` and `q` is explained in Table 11.3.1. The matrices `y` and `q` can be used to compute the forced response by function `respCV` (see Window 3.5).

In executing `forcedMD`, decoupled equations about modal coordinates like Eq. (11.26) are displayed in the MATLAB command window as a midstep in modal analysis.

The `y` and `q` in Windows 3.1 and 3.2 are $n_y \times 6 \times n$ and $n_q \times 6 \times n$ matrices representing the displacement vector $\{x(t)\}$ and the modal displacement vector $\{q(t)\}$, respectively. Here n is the number of degrees of freedom of the system, n_y and n_q are the maximum numbers of terms in each of the components of $\{x(t)\}$ and $\{q(t)\}$, respectively. The jth displacement $x_j(t)$ and the kth modal displacement $q_k(t)$ are expressed by the submatrices

$$y(:,:,j) = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n_y} \end{bmatrix} \in R^{n_y \times 6}, \quad q(:,:,k) = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n_q} \end{bmatrix} \in R^{n_q \times 6} \tag{11.44}$$

where α_l and β_l are one-by-six row vectors, for which the format and corresponding mathematical expression are given in Table 11.3.1. For instance, the matrix

$$y(:,:,2) = \begin{bmatrix} 1 & -0.25 & 0 & 0 & 0 & 0 \\ 5 & 0.264 & -0.178 & -0.359 & 6.785 & 0 \\ 4 & 0.867 & -0.051 & 1 & 0 & 0 \end{bmatrix}$$

TABLE 11.3.1 The Format of the Vectors in Eq. (11.44)

Vectors α_l and β_l	*Mathematical Expression*
[0 0 0 0 0 0]	$z(t) = 0$
[1 a 0 0 0 0]	$z(t) = a$
[2 a m 0 0 0]	$z(t) = a\,t^m$
[3 a σ 0 0 0]	$z(t) = a\,e^{\sigma t}$
[4 a σ m 0 0]	$z(t) = a\,t^m e^{\sigma t}$
[5 a b σ ω 0]	$z(t) = e^{\sigma t}(a\cos\omega t + b\sin\omega t)$
[6 a b σ ω m]	$z(t) = t^m e^{\sigma t}(a\cos\omega t + b\sin\omega t)$

indicates that

$$x_2(t) = -0.25 + e^{-0.359t}(0.264\cos 6.785t - 0.178\sin 6.785t) + 0.867te^{-0.051t}.$$

The meaning of y and q given in Table 11.3.1 is automatically interpreted and displayed in the MATLAB command window when functions `freeMD` and `forcedMD` are used.

External forces are specified according to Table 11.3.2. The last row of the table is about a user-specified forcing function $f(t)$, for which `Load_Spec` is a two-row matrix containing the coefficients of the numerator and denominator of the Laplace transform $F(s) = \mathcal{L}[f(t)]$. For instance,

$$\texttt{Load\_Spec} = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

implies

$$F(s) = \frac{s+2}{3s^2+4s+5} \quad \text{or} \quad f(t) = \mathcal{L}^{-1}[F(s)] = 0.58346e^{-0.83333t}\sin 1.1426t.$$

If a system is under both initial and external disturbances, or under multiple external loads, use functions `freeMD` and `forcedMD` by the superposition principle; see Section 11.3.5.

TABLE 11.3.2 Load Specification Vector `Load_Spec` for Functions `forcedMD` and `forcedLT`

External Load	*f(t)*	`Load_Spec`
Impulse load	$I_0\delta(t)$, where $\delta(t)$ is the delta function	$[0\ I_0]$
Step load	F_0	$[1\ F_0]$
Ramp load	$a_0 t$	$[2\ a_0]$
Exponential load	$A\,e^{bt}$	$[3\ A\ b]$
Sinusoidal load	$A\sin(\omega t + \phi_0)$	$[4\ A\ \omega\ \phi_0]$ ω in rad/s, ϕ_0 in degrees
User-specified load	$f(t) = \mathcal{L}^{-1}[F(s)]$ where $\mathcal{L}^{-1}$ is the inverse Laplace operator, $F(s) = \dfrac{b_p s^p + b_{p-1}s^{p-1} + \cdots + b_1 s + b_0}{a_p s^p + a_{p-1}s^{p-1} + \cdots + a_1 s + a_0}$ and $a_p \neq 0$.	$\begin{bmatrix} b_p & b_{p-1} & \cdots & b_1 & b_0 \\ a_p & a_{p-1} & \cdots & a_1 & a_0 \end{bmatrix}$

EXAMPLE 3.1

Consider a 3-DOF system with the inertia and stiffness matrices

$$[M] = \begin{bmatrix} 75 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 600 \end{bmatrix}, \quad [K] = \begin{bmatrix} 10 & -10 & 0 \\ -10 & 40 & -20 \\ 0 & -20 & 50 \end{bmatrix}$$

and with damping ratio of 0.1 for every mode of vibration. Let the initial conditions be $\{x(0)\} = (0 \quad 0.2 \quad 0)^T$ and $\{\dot{x}(0)\} = (0 \quad 0 \quad 0)^T$. The free response of the system is obtained by the commands

```
>> m=diag([75 100 600]); k=10*[1 -1 0; -1 4 -2; 0 -2 5];
>> setmdof(m, k, [2 0.1])
>> x0 = [0 0.2 0]'; v0 = [0 0 0]';
>> freeMD(x0,v0)
```

which yield the following results:

```
Eigenvalues:
   -0.0221 + 0.2199i -0.0221 - 0.2199i
   -0.0330 + 0.3286i -0.0330 - 0.3286i
   -0.0677 + 0.6739i -0.0677 - 0.6739i

Normalized eigenvectors [U1 U2 ...]:
   0.0542  0.0947  0.0378
   0.0344  0.0172 -0.0923
   0.0332 -0.0223  0.0082

Modal (decoupled) equations:

   (d/dt)^2*q1(t) + 0.044204*d/dt*q1(t) + 0.04885*q1(t) = 0
   subject to: q1(0) = 0.68703, (d/dt)*q1(0) = 0

   (d/dt)^2*q2(t) + 0.066055*d/dt*q2(t) + 0.10908*q2(t) = 0
   subject to: q2(0) = 0.34438, (d/dt)*q2(0) = 0

   (d/dt)^2*q3(t) + 0.13546*d/dt*q3(t) + 0.45873*q3(t) = 0
   subject to: q3(0) = -1.8465, (d/dt)*q3(0) = 0

Modal coordinates:

   q1(t) = exp(-0.022102*t)*[0.68703*cos(0.21991*t)
           + (0.069049)*sin(0.21991*t)]

   q2(t) = exp(-0.033028*t)*[0.34438*cos(0.32862*t)
           + (0.034612)*sin(0.32862*t)]

   q3(t) = exp(-0.06773*t)*[-1.8465*cos(0.6739*t)
           + (-0.18558)*sin(0.6739*t)]
```

```
Displacements in physical coordinates

x1(t) = exp(-0.022102*t)*[0.037246*cos(0.21991*t)
        + (0.0037434)*sin(0.21991*t)]
        + exp(-0.033028*t)*[0.032604*cos(0.32862*t)
        + (0.0032768)*sin(0.32862*t)]
        + exp(-0.06773*t)*[-0.06985*cos(0.6739*t)
        + (-0.0070202)*sin(0.6739*t)]

x2(t) = exp(-0.022102*t)*[0.0236*cos(0.21991*t)
        + (0.0023719)*sin(0.21991*t)]
        + exp(-0.033028*t)*[0.0059299*cos(0.32862*t)
        + (0.00059598)*sin(0.32862*t)]
        + exp(-0.06773*t)*[0.17047*cos(0.6739*t)
        + (0.017133)*sin(0.6739*t)]

x3(t) = exp(-0.022102*t)*[0.022813*cos(0.21991*t)
        + (0.0022928)*sin(0.21991*t)]
        + exp(-0.033028*t)*[-0.0076764*cos(0.32862*t)
        + (-0.0007715)*sin(0.32862*t)]
        + exp(-0.06773*t)*[-0.015137*cos(0.6739*t)
        + (-0.0015213)*sin(0.6739*t)]
```

Each of the modal coordinates $q_k(t)$ is only related to a single mode, while each of the physical displacements $x_j(t)$ involves all the three modes.

Consider a step load $f(t) = 15$ that is applied to the first mass, namely $\{B_f\} = \begin{pmatrix}1 & 0 & 0\end{pmatrix}^T$. Let the system be at rest initially. By

```
>> load = [1 15];
>> forcedMD(load)
```

the forced response is obtained as follows:

```
Modal (decoupled) equations:

 (d/dt)^2*q1(t) + 0.044204*d/dt*q1(t) + 0.04885*q1(t) =
0.054214*f(t)

 (d/dt)^2*q2(t) + 0.066055*d/dt*q2(t) + 0.10908*q2(t) =
0.094674*f(t)

 (d/dt)^2*q3(t) + 0.13546*d/dt*q3(t) + 0.45873*q3(t) =
0.037829*f(t)

Modal coordinates:

 q1(t) = 16.6471 + exp(-0.022102*t)*[-16.6471*cos(0.21991*t)
         + (-1.6731)*sin(0.21991*t)]

 q2(t) = 13.0186 + exp(-0.033028*t)*[-13.0186*cos(0.32862*t)
         + (-1.3084)*sin(0.32862*t)]

 q3(t) = 1.237 + exp(-0.06773*t)*[-1.237*cos(0.6739*t)
         + (-0.12432)*sin(0.6739*t)]
```

```
Displacements in physical coordinates

x1(t) = 2.1818 + exp(-0.022102*t)*[-0.9025*cos(0.21991*t)
       + (-0.090705)*sin(0.21991*t)]
       + exp(-0.033028*t)*[-1.2325*cos(0.32862*t)
       + (-0.12387)*sin(0.32862*t)]
       + exp(-0.06773*t)*[-0.046794*cos(0.6739*t)
       + (-0.004703)*sin(0.6739*t)]

x2(t) = 0.68182 + exp(-0.022102*t)*[-0.57185*cos(0.21991*t)
       + (-0.057473)*sin(0.21991*t)]
       + exp(-0.033028*t)*[-0.22417*cos(0.32862*t)
       + (-0.02253)*sin(0.32862*t)]
       + exp(-0.06773*t)*[0.1142*cos(0.6739*t)
       + (0.011478)*sin(0.6739*t)]

x3(t) = 0.27273 + exp(-0.022102*t)*[-0.55278*cos(0.21991*t)
       + (-0.055556)*sin(0.21991*t)]
       + exp(-0.033028*t)*[0.29019*cos(0.32862*t)
       + (0.029165)*sin(0.32862*t)]
       + exp(-0.06773*t)*[-0.01014*cos(0.6739*t)
       + (-0.0010191)*sin(0.6739*t)]
```

11.3.2 Laplace Transform Method

In this section, Eq. (11.22) is solved by another analytical method, i.e., the Laplace transform method. Laplace transform of Eq. (11.22) with respect to time gives

$$\left(s^2[M] + s[C] + [K]\right)\{X(s)\} = [M](s\{x_0\} + \{v_0\}) + [C]\{x_0\} + \{F(s)\} \tag{11.45}$$

where s is the complex Laplace transform parameter, and $\{X(s)\}$ and $\{F(s)\}$ are the Laplace transforms of $\{x(t)\}$ and $\{F(t)\}$, respectively. It follows that

$$\{X(s)\} = \left(s^2[M] + s[C] + [K]\right)^{-1}(s[M]\{x_0\} + [M]\{v_0\} + [C]\{x_0\} + \{F(s)\}) \tag{11.46}$$

Thus, the time response of the system is given by

$$\{x(t)\} = \mathcal{L}^{-1}[\{X(s)\}] = \{x(t)\}_{Free} + \{x(t)\}_{Forced} \tag{11.47}$$

where $\mathcal{L}^{-1}$ is the inverse Laplace transform operator, and

$$\begin{aligned} \{x(t)\}_{Free} &= \mathcal{L}^{-1}\left[\left(s^2[M] + s[C] + [K]\right)^{-1}(s[M]\{x_0\} + [M]\{v_0\} + [C]\{x_0\})\right] \\ \{x(t)\}_{Forced} &= \mathcal{L}^{-1}\left[\left(s^2[M] + s[C] + [K]\right)^{-1}\{F(s)\}\right] \end{aligned} \tag{11.48}$$

Obviously, $\{x(t)\}_{Free}$ is the free response of the system, and $\{x(t)\}_{Forced}$ is the forced response of the system (under zero initial conditions). The above formulas are valid for both proportionally and nonproportionally damped system.

Solution by the Toolbox Although the formulas in Eqs. (11.47) and (11.48) are compact, the inverse Laplace transforms involved are difficult to obtain, even for a system with a few degrees of freedom. For this reason, Laplace transform method is mainly used for analytical derivations. The Toolbox of this chapter provides functions `freeLT` and `forcedLT` that can obtain explicit mathematical expressions of free and forced response of an M-DOF system by inverse Laplace transform; see Windows 3.3 and 3.4. To be able to use functions `freeLT` and `forcedLT`, the MATLAB Control System Toolbox is needed.

Window 3.3. Function freeLT

MATLAB Function: **freeLT**

Purpose:
To obtain the mathematical expressions of free response of an M-DOF system by inverse Laplace transform

$$\{x(t)\}_{Free} = \mathcal{L}^{-1}\left[\left(s^2[M] + s[C] + [K]\right)^{-1}\left(s[M]\{x_0\} + [M]\{v_0\} + [C]\{x_0\}\right)\right]$$

The system can be nonproportionally damped.

Synopsis:

```
freeLT(x0, v0)
y = freeLT(x0, v0)
```

Description:
`freeLT(x0, v0)` displays the mathematical expressions of free response of the system subject to initial disturbances, where `x0` is a vector of initial displacements and `v0` a vector of initial velocities.

`y = freeLT(x0, v0)` returns the coefficients of the mathematical expressions of the free response $\{x(t)\}$ in a three-dimensional matrix `y`. The meaning of the entries of `y` is explained in Table 11.3.1. The matrix `y` can be used to compute the free response by function `respCV` (see Window 3.5).

Window 3.4. Function forcedLT

MATLAB Function: **forcedLT**

Purpose:
To obtain the mathematical expressions of forced response of an M-DOF system by inverse Laplace transform

$$\{x(t)\}_{Forced} = \mathcal{L}^{-1}\left[\left(s^2[M] + s[C] + [K]\right)^{-1}\{F(s)\}\right].$$

Here zero initial conditions are imposed. The system can be nonproportionally damped.

Window 3.4. Function forcedLT (*Continued*)

Synopsis:

```
forcedLT(Load_Spec, Bf)
y = forcedLT(Load_Spec, Bf)
```

Description:

`forcedLT(Load_Spec, Bf)` displays the mathematical expressions of force response of the system subject to an external force $f(t)$ specified by a vector or matrix `Load_Spec`, where `Bf` is the load distribution vector. By default `Bf` $= (1 \quad 0 \quad 0 \quad \cdots \quad 0)^T$. Specification of vector `Load_Spec` is given in Table 11.3.2.

`y = forcedLT(Load_Spec, Bf)` returns the coefficients of the mathematical expressions of the forced response $\{x(t)\}$ in a three-dimensional matrix `y`. The meaning of the entries of `y` is explained in Table 11.3.1. The matrix `y` can be used to compute the forced response by function `respCV` (see Window 3.5).

Note 1. Tables 11.3.1 and 11.3.2 are valid for functions `freeLT` and `forcedLT`.

Note 2. Functions `freeLT` and `forcedLT` are applicable to a system under both initial disturbances and multiple external loads; see Section 11.3.5.

Note 3. Unlike functions `freeMD` and `forceMD` in modal analysis, `freeLT` and `forcedLT` are valid for nonproportionally damped systems.

EXAMPLE 3.2

Consider a vibrating system of three degrees of freedom

$$\begin{bmatrix} 7 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 30 \end{bmatrix} \{\ddot{x}(t)\} + \begin{bmatrix} 4 & -2 & 0 \\ -2 & 6 & -4 \\ 0 & -4 & 10 \end{bmatrix} \{\dot{x}(t)\} + \begin{bmatrix} 10 & -10 & 0 \\ -10 & 30 & -20 \\ 0 & -20 & 60 \end{bmatrix} \{x(t)\} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} f(t)$$

which is a nonproportionally damped system considered in Example 2.2. Assume that the system is subject to the initial disturbances

$$\{x(0)\} = (0 \quad 0 \quad -0.5)^T \quad \text{and} \quad \{\dot{x}(0)\} = (0 \quad 0 \quad 0)^T.$$

The free response of the system is obtained by

```
>> M = diag([7 10 30]); K = [10 -10 0; -10 30 -20; 0 -20 60];
>> C = [4 -2 0; -2 6 -4; 0 -4 10];
>> setmdof(M, K, C)
>> x0 = [0 0 -0.5]'; v0 = [0 0 0]';
>> freeLT(x0, v0)
```

which yields the following:

```
x1(t) = exp(-0.12176*t)*[-0.33961*cos(0.71904*t)
        + (0.06139)*sin(0.71904*t)]
        + exp(-0.20442*t)*[0.5132*cos(1.3011*t)
        + (-0.022056)*sin(1.3011*t)]
        + exp(-0.4262*t)*[-0.17359*cos(1.9869*t)
        + (-0.013022)*sin(1.9869*t)]

x2(t) = exp(-0.12176*t)*[-0.2079*cos(0.71904*t)
        + (0.064545)*sin(0.71904*t)]
        + exp(-0.20442*t)*[-0.1029*cos(1.3011*t)
        + (-0.10169)*sin(1.3011*t)]
        + exp(-0.4262*t)*[0.3108*cos(1.9869*t)
        +(0.086574)*sin(1.9869*t)]

x3(t) = exp(-0.12176*t)*[-0.090254*cos(0.71904*t)
        + (0.038488)*sin(0.71904*t)]
        + exp(-0.20442*t)*[-0.31709*cos(1.3011*t)
        + (-0.057052)*sin(1.3011*t)]
        + exp(-0.4262*t)*[-0.092652*cos(1.9869*t)
        + (-0.034598)*sin(1.9869*t)]
```

Now, a sinusoidal force $f(t) = 3\sin(0.72t)$ is applied to the first station. By

```
>> Load_Spec = [4 3 0.72 0];
>> forcedLT(Load_Spec)
```

the forced response is obtained as follows:

```
x1(t) = exp(-0.12176*t)*[1.2786*cos(0.71904*t)
        + (0.32328)*sin(0.71904*t)] -1.2873*cos(0.72*t)
        + (0.038648)*sin(0.72*t)
        + exp(-0.20442*t)*[0.0081298*cos(1.3011*t)
        + (-0.069999)*sin(1.3011*t)]
        + exp(-0.4262*t)*[0.00062329*cos(1.9869*t)
        + (-0.0058354)*sin(1.9869*t)]

x2(t) = exp(-0.12176*t)*[0.82531*cos(0.71904*t)
        + (0.10409)*sin(0.71904*t)] -0.80606*cos(0.72*t)
        + (-0.020703)*sin(0.72*t)
        + exp(-0.20442*t)*[-0.016005*cos(1.3011*t)
        + (0.011737)*sin(1.3011*t)]
        + exp(-0.4262*t)*[-0.0032476*cos(1.9869*t)
        + (0.010381)*sin(1.9869*t)]

x3(t) = exp(-0.12176*t)*[0.37484*cos(0.71904*t)
        + (0.0087742)*sin(0.71904*t)] -0.36151*cos(0.72*t)
        + (-0.015646)*sin(0.72*t)
        + exp(-0.20442*t)*[-0.014598*cos(1.3011*t)
        + (0.04172)*sin(1.3011*t)]
        + exp(-0.4262*t)*[0.0012643*cos(1.9869*t)
        + (-0.0030853)*sin(1.9869*t)]
```

As can be seen, the steady-state response of the system (after a long time) is

$$\{x(t)\}_{ss} = -\begin{pmatrix}1.2873\\0.80606\\0.36151\end{pmatrix}\cos(0.72t) + \begin{pmatrix}0.038648\\-0.020703\\-0.015646\end{pmatrix}\sin(0.72t)$$

11.3.3 Plotting Analytical Solutions

Quite often, the analytical vibration solutions obtained by modal analysis (Section 11.3.1) and Laplace transform (Section 11.3.2) need to be plotted against time. As shown in Examples 3.1 and 3.2, the expressions of an analytical solution are usually long and complicated even for a relatively simple system. To resolve this problem, the Toolbox provides a function `respCV`, which automatically converts analytical solutions into numerical data that can be used to plot response curves; see Window 3.5.

Window 3.5. Function respCV

MATLAB Function: **respCV**

Purpose:
To compute the response of an M-DOF system based on the mathematical expressions obtained by functions `freeMD` and `forcedMD` (modal analysis, Section 11.3.1), or by functions `freeLT` and `forcedLT` (Laplace transform method, Secion 11.3.2).

Synopsis:

```
[y, t] = respCV(y_analyt)
[y, t] = respCV(y_analyt, tf, npts, Td)
y = respCV(y_analyt, tf, npts, Td)
```

Description:
`[y, t] = respCV(y_analyt)` returns the data of system response in an `n`-row matrix `y` where `n` is the number of degrees of freedom of the system, and the times of computation in a vector `t`. Here, matrix `y` is computed based on the analytical solution `y_analyt` that is obtained by one of the functions `freeMD`, `forcedMD`, `freeLT`, and `forcedLT`. With `y` and `t`, the system response can be plotted by the MATLAB function `plot`. The `y(j,:)` represents the jth displacement of the system. For instance, the commands

```
[ya, qa] = freeMD(x0,v0);
q = respCV(qa);
plot(t, q(2,:))
```

plot the second modal coordinate $q_2(t)$ versus time in free vibration. The function `respCV` itself automatically plots up to three response curves.

`[y, t] = respCV(y_analyt, tf, npts, Td)` computes the the above-mentioned system response with the additional setting:

`tf`—total computation time;
`npts`—number of time points in computation;
`Td`—time shift (time delay) of the time history of system response.

By default, `tf` = 4T, where T is the largest period of the system, `npts` = 501, and `Td` = 0. Any one of the parameters `tf`, `npts`, and `Td` can be replaced by `[]` as a placeholder for its default value.

`y = respCV(y_analyt, tf, npts, Td)` returns the computed system response `y` only. With `y`, the system response can be plotted by function `plotvib(y)` (see Window 3.8).

The time delay parameter `Td` in function `respCV` allows the user to shift the time history of system response as follows

$$\{x(t - T_d)\} = \begin{cases} 0 & \text{for } t < T_d \\ \{x(t - T_d)\} & \text{for } t \geq T_d \end{cases} \tag{11.49}$$

This feature is useful for evaluating system response subject to an external load that is not applied initially, but sometime later.

EXAMPLE 3.3

Consider the system in Example 3.1. To plot the free response of the system subject to the initial conditions $\{x(0)\} = (0.1 \quad -0.2 \quad 0.3)^T$ and $\{\dot{x}(0)\} = (0 \quad 0 \quad 0)^T$, type

```
>> m=diag([75 100 600]);
>> k=10*[1 -1 0; -1 4 -2; 0 -2 5];
>> setmdof(m,k, [2 0.1])
>> x0 = [0.1 -0.2 0.3]';
>> v0 = [0 0 0]';
>> [ya, qa] = freeMD(x0, v0);
>> [x, t] = respCV(ya);
>> plot(t,x(1,:),t,x(2,:),':', t,x(3,:), '-.'); grid
>> xlabel('time, t');
>> legend('x1', 'x2', 'x3')
```

which gives the response curves in Fig. 11.3.1. To plot the third modal coordinate $q_3(t)$, type

```
>> [q, t] = respCV(qa);
>> plot(t,q(3,:)); grid
>> xlabel('time, t'); ylabel('q_3(t)')
```

which yields Fig. 11.3.2. Assume that the system is subject to a step function $f(t) = 12.5$ applied to the third station at time $t = 30$. By

```
>> load = [1 12.5]; Bf = [0 0 1]';
>> ya = forcedMD(load, Bf);
>> [y, t] = respCV(ya, [], [], 30);
```

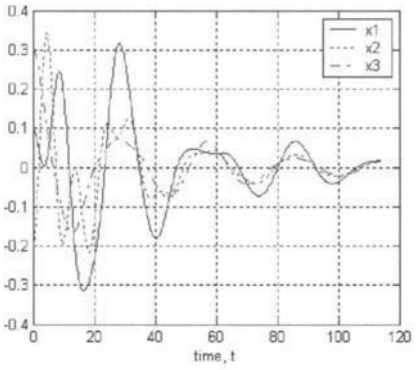


FIGURE 11.3.1

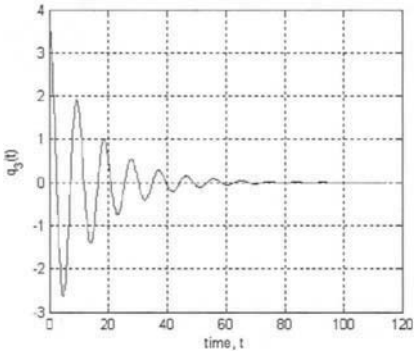


FIGURE 11.3.2

```
>> plot(t,y(1,:),t,y(2,:),':', t,y(3,:), '-.'); grid
>> xlabel('time, t');
>> legend('x1', 'x2', 'x3')
```

the forced response of the system is plotted in Fig. 11.3.3. Here the time shift feature of function `respCV` (`Td` = 30) has been used.

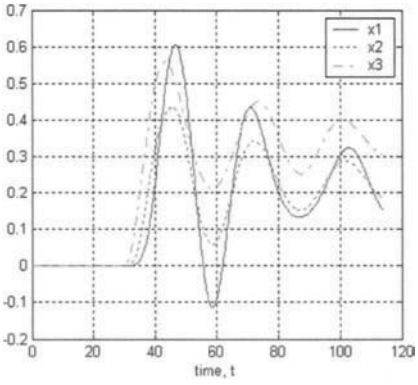


FIGURE 11.3.3

11.3.4 Runge-Kutta Algorithm

The vibration problem of an M-DOF system is solved via numerical integration. To this end, the original second-order equation of motion (11.22) is cast to the first-order state equation

$$\{\dot{z}(t)\} = \Phi(t,\ \{z(t)\}), \quad \{\dot{z}(0)\} = \{z_0\} \tag{11.50}$$

where

$$\{z(t)\} = \begin{Bmatrix} \{x(t)\} \\ \{\dot{x}(t)\} \end{Bmatrix}, \quad \{z_0\} = \begin{Bmatrix} \{x_0\} \\ \{v_0\} \end{Bmatrix}$$
$$\Phi(t, \{z(t)\}) = [A]\{z(t)\} + [B]f(t) \tag{11.51}$$

and matrices $[A]$ and $[B]$ are given in Eq. (11.35). The state equation (11.50) is solved by the following fixed-step Runge-Kutta method of order four:

$$\{z_{k+1}\} = \{z_k\} + \frac{h}{6}(\{f_1\} + 2\{f_2\} + 2\{f_3\} + \{f_4\}) \tag{11.52}$$

where

$$\begin{aligned} \{f_1\} &= \Phi(t_k, \{z_k\}) \\ \{f_2\} &= \Phi\left(t_k + \frac{h}{2}, \{z_k\} + \frac{h}{2}\{f_1\}\right) \\ \{f_3\} &= \Phi\left(t_k + \frac{h}{2}, \{z_k\} + \frac{h}{2}\{f_2\}\right) \\ \{f_4\} &= \Phi(t_k + h, \{z_k\} + h\{f_3\}) \end{aligned} \tag{11.53}$$

$\{z_k\} = \{z(t_k)\}$, $t_k = kh$, $k = 0, 1, 2, \ldots$, and h is the step size. In an advanced algorithm, the step size h is adaptively adjusted during the computation. The software MATLAB provides Runge-Kutta solvers like `ode45` with automatic step-size adjustment capability.

Solution by the Toolbox The Toolbox of this chapter has two functions `freeRK` and `forcedRK` for computing the response of an M-DOF system by the above Runge-Kutta algorithm. These functions are described in Windows 3.6 and 3.7.

Window 3.6. Function freeRK

MATLAB Function: **freeRK**

Purpose:
To compute the free response of an M-DOF system by the Runge-Kutta algorithm described in Eqs. (11.52) and (11.53).

Synopsis:

```
freeRK(x0, v0)
[x, t, v] = freeRK(x0, v0, tf, npts)
```

Window 3.6. Function freeRK (*Continued*)

Description:
`freeRK(x0, v0)` plots the free response of the system versus time, where `x0` is a vector of initial displacements and `v0` a vector of initial velocities.

`[x, t, v] = freeRK(x0, v0, tf, npts)` returns the computed displacements and velocities of the system in n-row matrices `x` and `v`, respectively, and the times of computation in a vector `t`. Here `tf` is total computation time and `npts` the number of times in the region $0 \leq t \leq$ `tf`. By default, `tf` = 4T, where T is the largest period of the system, and `npts` = 501. The parameter `tf` can be replaced by `[]` as a placeholder for its default value, such as `[x, t, v] = freeRK(x0, v0, [], npts)`.

Window 3.7. Function forcedRK

MATLAB Function: **forcedRK**

Purpose:
To compute the response of an M-DOF system under an external force and zero initial disturbances by the Runge-Kutta algorithm described in Eqs. (11.52) and (11.53).

Synopsis:

```
forcedRK(Load_Spec, Bf)
[x, t, v] = forcedRK(Load_Spec, Bf, tf, npts)
```

Description:
`forcedRK(Load_Spec, Bf)` plots the forced response versus time, where `Load_Spec` is a vector specifying the external load by Table 11.3.3, and `Bf` is the load distribution vector.

`[x, t, v] = forcedRK(Load_Spec, Bf, tf, npts)` returns the computed displacements and velocities of the system in n-row matrices `x` and `v`, respectively, and the times of computation in a vector `t`. Here `tf` is total computation time and `npts` the number of times in the region $0 \leq t \leq$ `tf`. By default, `tf` = 4T, where T is the largest period of the system, and `npts` = 501. Also, `tf` can be replaced by `[]` as a placeholder for its default value, such as `[x, t, v] = forcedRK(Load_Spec, Bf, [], npts)`.

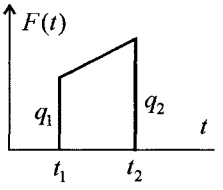
Note 1. The output arguments `x`, `v`, and `t` of functions `freeRK` and `forcedRK` can be used to plot response curves as follows:

`plot(t, x(j, ;))`	or	`plotvib(x, j)`	plot the jth displacement versus time
`plot(t, v(j, ;))`	or	`plotvib(v, j)`	plot the jth velocity versus time

Here `plotvib` is a plotting function from the Toolbox; see Window 3.8.

Note 2. For each of the first five loads listed in Table 11.3.3, there is a time-shifted or delayed forcing function, which is expressed as $F(t - T_d)\,u(t - T_d)$, where $u(t - T_d)$ is a shifted unit

TABLE 11.3.3 Specification of Vector Load_Spec for Function forcedRK

External Force	$f(t)$	Load_Spec
Impulse force	$I_0\delta(t)$ $I_0\delta(t-T_d)$	$[0\ I_0]$ $[0\ I_0\ T_d]$
Step force	F_0 $F_0u(t-T_d)$	$[1\ F_0]$ $[1\ F_0\ T_d]$
Ramp force	$a_0\,t$ $a_0\,(t-T_d)\,u(t-T_d)$	$[2\ a_0]$ $[2\ a_0\ T_d]$
Exponential force	$A\,e^{bt}$ $A\,e^{b(t-T_d)}\,u(t-T_d)$	$[3\ A\ b]$ $[3\ A\ b\ T_d]$
Sinusoidal force	$A\sin(\omega t+\phi_0)$ $A\sin(\omega(t-T_d)+\phi_0)\ u(t-T_d)$	$[4\ A\ \omega\ \phi_0]$ ω in rad/s, ϕ_0 in degrees $[4\ A\ \omega\ \phi_0\ T_d]$ ω in rad/s, ϕ_0 in degrees
Pulse force		$[5\ t_1\ \ q_1\ \ t_2\ \ q_2]$

step function given by

$$u(t-T_d)=\begin{cases}1 & \text{for } t\geq T_d\\ 0 & \text{for } t<T_d\end{cases}$$

These delayed functions are useful for describing those external forces that are applied after the initial time.

EXAMPLE 3.4

Consider the system in Example 3.2, which is set up by

```
>> M = diag([7 10 30]);
>> K = [10 -10 0; -10 30 -20; 0 -20 60];
>> C = [4 -2 0; -2 6 -4; 0 -4 10];
>> setmdof(M, K, C);
```

Let the initial conditions be

$$\{x(0)\}=\begin{pmatrix}0.2 & 0 & 0.1\end{pmatrix}^T,\quad \{\dot{x}(0)\}=\begin{pmatrix}0 & 0 & 0\end{pmatrix}^T.$$

By the commands

```
>> x0 = [0.2 0 0.1]'; v0 = [0 0 0]';
>> freeRK(x0,v0)
```

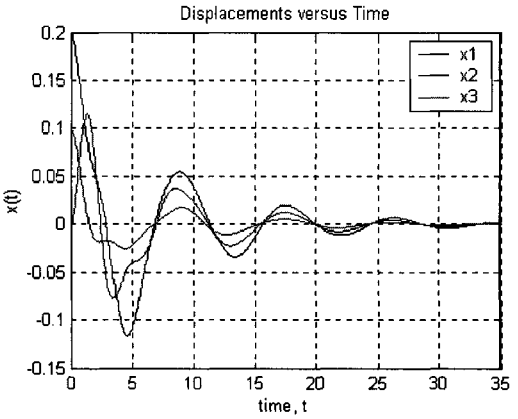


FIGURE 11.3.4

the free response of the system is plotted in Fig. 11.3.4. If a pulse force ($t_1 = 10$, $q_1 = -5$, $t_2 = 20$, and $q_1 = 12$) is applied to the third station, the forced response is obtained by

```
>> load = [5 10 -5 20 12]; Bf = [0 0 1]';
>> forcedRK(load, Bf)
```

See Fig. 11.3.5 for the plotted response.

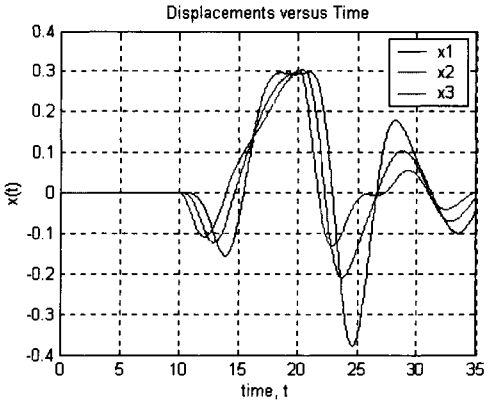


FIGURE 11.3.5

11.3.5 Solution by Superposition

Now consider a system subject to multiple external loads

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \sum_{j=1}^{n_f} \{B_j\} f_j(t), \quad t \geq 0$$
$$\{x(0)\} = \{x_0\}, \quad \{\dot{x}(0)\} = \{v_0\} \tag{11.54}$$

where $f_j(t)$ is the jth external load and $\{B_j\}$ is the corresponding force distribution vector. The total response of the system is expressed as

$$\{x(t)\} = \{x(t)\}_{Free} + \{x(t)\}_1 + \{x(t)\}_2 + \cdots + \{x(t)\}_{n_f} \tag{11.55}$$

where $\{x(t)\}_{Free}$ is the free response of the system due to the initial disturbances $\{x_0\}$ and $\{v_0\}$, and $\{x(t)\}_j$ is the forced response due to the jth external load $f_j(t)$.

By the Toolbox, the free response $\{x(t)\}_{Free}$ can be obtained by functions `freeMD`, `freeLT`, and `freeRK`; the forced response $\{x(t)\}_j$ can be determined by functions `forcedMD`, `forcedLT`, and `forcedRK`. The procedure of solution by superposition is summarized below, in which $\{B_j\} f_j(t)$, $j = 1, 2, \ldots, n_f$, are specified by vectors `Load_1`, `Load_2`, ..., `Load_nf`, and vectors `Bf_1`, `Bf_2`, ..., `Bf_nf`, respectively.

Solution by Modal Analysis

Step 1. Determination of free response

```
ya0 = freeMD(x0, v0); z0 = respCV(ya0);
```

Step 2. Determination of forced response

```
ya_1 = forcedMD(Load_1, Bf_1); [z_1, t] = respCV(ya_1);
ya_2 = forcedMD(Load_2, Bf_2); z_2 = respCV(ya_2);
. . . . . .
ya_nf = forcedMD(Load_nf, Bf_nf); z_nf = respCV(ya_nf);
```

Step 3. Solution by superposition

```
x = z0 + z_1 + z_2 + ··· + z_nf;
```

To plot total response, use either `plotvib(x)` or `plot(t,x)`

Solution by Laplace Transform

Step 1. Determination of free response

```
ya0 = freeLT(x0, v0); z0 = respCV(ya0);
```

Step 2. Determination of forced response

```
ya_1 = forcedLT(Load_1, Bf_1); [z_1, t] = respCV(ya_1);
ya_2 = forcedLT(Load_2, Bf_2); z_2 = respCV(ya_2);
. . . . . .
ya_nf = forcedLT(Load_nf, Bf_nf); z_nf = respCV(ya_nf);
```

Step 3. Solution by superposition

```
x = z0 + z_1 + z_2 + ··· + z_nf;
```

To plot total response, use either `plotvib(x)` or `plot(t,x)`

Solution by Runge-Kutta Numerical Integration

Step 1. Determination of free response

```
z0 = freeRK(x0, v0);
```

Step 2. Determination of forced response

```
[z_1, t] = forcedRK(Load_1, Bf_1);
z_2 = forcedRK(Load_2, Bf_2);
. . . . . .
z_nf = forcedRK(Load_nf, Bf_nf);
```

Step 3. Solution by superposition

```
x = z0 + z_1 + z_2 + ··· + z_nf;
```

To plot total response, use either `plot(t,x)` or `plotvib(x)`

Note 1. In solution via modal analysis or Laplace transform, function `respCV` can be called to convert mathematical expressions to numerical data, as shown in Window 3.5.

Note 2. `plotvib` is a function from the Toolbox, which plots system response obtained by functions `respCV`, `freeRK`, or `forcedRK`; see Window 3.8.

Note 3. Functions `respCV`, `freeRK`, and `forcedRK` pass times of simulation internally to function `plotvib`. So, using `plotvib` does not need to output the time vector `t`. For instance, the commands

```
[z_1, t] = respCV(ya_1);
plot(t, z_1)
```

can be replaced by

```
z_1 = respCV(ya_1);
plotvib(z_1)
```

which produces the same result.

Window 3.8. Function `plotvib`

MATLAB Function: **`plotvib`**

Purpose:
To plot the response of an M-DOF system versus time.

Window 3.8. Function `plotvib` (*Continued*)

Synopsis:

```
plotvib(y, index)
```

Description:

`plotvib(y, index)` plots the displacements or velocities of the system versus time by using an `n`-row matrix `y` that is obtained by functions `respCV`, `freeRK`, `forcedRK`, `ctrmdof1`, and `ctrmdof2`, where `index` is a row vector of integers indicating which displacements or velocities to plot. For instance, the commands

```
y = freeRK(x0,v0);
plotvib(y, [1 4])
```

plot the free displacements (`y(1,:)` and `y(4,:)`) at the first and fourth stations that are computed by the Runge-Kutta algorithm. By default, `index` = 1.

EXAMPLE 3.5

For the 2-DOF system (nonproportionally damped)

$$\begin{bmatrix} 5 & 0 \\ 0 & 18 \end{bmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{bmatrix} 4 & -2 \\ -2 & 6 \end{bmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \begin{bmatrix} 25 & -10 \\ -10 & 50 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2\cos(1.5t) \\ -20\delta(t-6) \end{pmatrix}$$

$$\begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix} = \begin{pmatrix} 0.4 \\ -0.2 \end{pmatrix}, \quad \begin{pmatrix} \dot{x}_1(0) \\ \dot{x}_2(0) \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

its total response is computed and plotted by the following commands

```
% system setup
>> m=diag([5 18]); c=[4 -2; -2 6]; k = [25 -10; -10 50];
>> setmdof(m, k, c)
% free response
>> ya0 = freeLT([0.4 -0.2]', [1 -1]'); z0 = respCV(ya0, 20);
% forced response due to a sinusoidal load
>> ya1 = forcedLT([4 2 1.5 90], [1 0]'); z1 = respCV(ya1, 20);
% forced response due to a delayed impulse load
>> ya2 = forcedLT([0 -20], [0 1]'); z2 = respCV(ya2, 20, [], 6);
% plotting total response
>> plotvib(z0 + z1 + z2, [1 2])
```

See Fig. 11.3.6 for the plotted displacements. The impulse applied at $t = 6$ causes a sudden change of the direction of the second displacement x_2.

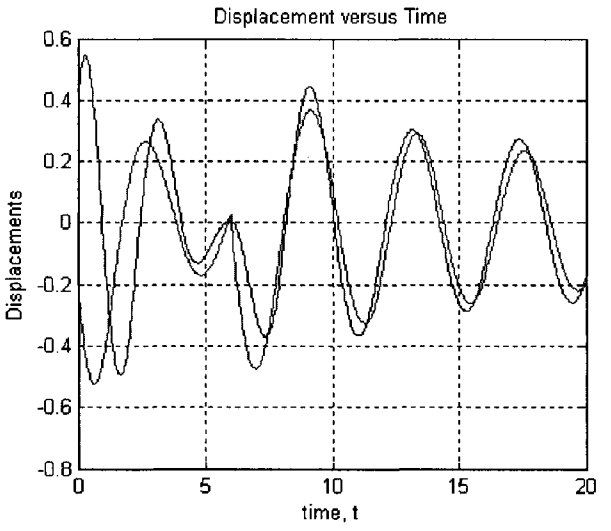


FIGURE 11.3.6 Total response.

11.3.6 Harmonic Excitation

Consider a system subject to a harmonic excitation

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \{B_f\} e^{j\omega t} \tag{11.56}$$

where ω is the excitation frequency and $j = \sqrt{-1}$. The steady-state solution of Eq. (11.56) is

$$\{x(t)\}_{ss} = \{X\} e^{j\omega t} \tag{11.57}$$

where

$$\{X\} = H(j\omega)\{B_f\} \tag{11.58}$$

with

$$H(j\omega) = \left(-\omega^2 [M] + j\omega [C] + [K]\right)^{-1} \tag{11.59}$$

Thus, the steady-state solution is expressed as

$$\{x(t)\}_{ss} = H(j\omega)\{B_f\} e^{j\omega t} \tag{11.60}$$

The matrix $H(j\omega)$ is called the *frequency response* of the system. Write

$$\{x(t)\}_{ss} = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix}, \quad \{X\} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} \tag{11.61}$$

The steady-state displacement at the kth station is of the form

$$x_k(t) = A_k e^{j(\omega t + \phi_k)} \tag{11.62}$$

with the magnitude and phase angle given by

$$A_k = |X_k|, \quad \phi_k = \arg(X_k) \tag{11.63}$$

The above results are valid if the complex harmonic factor $e^{j\omega t}$ is replaced by a real one like $\sin(\omega t + \theta_0)$.

Solution by the Toolbox The Toolbox of this chapter has functions `frfun` and `plotfr` for computing and plotting the magnitudes and phase angles of the steady-state response of an M-DOF system; see Windows 3.9 and 3.10.

Window 3.9. Function `frfun`

MATLAB Function: **`frfun`**

Purpose:
To determine the magnitudes and phase angles of steady-state displacements of an M-DOF system subject to a harmonic excitation.

Synopsis:

```
frfun(Bf, jx)
[y, freq] = frfun(Bf, jx, omg0, omgf, npts)
```

Description:
`frfun(Bf, jx)` plots the magnitude and phase of the steady-state displacement at station `jx`, where `Bf` is the force distribution vector $\{B_f\}$ in Eq. (11.56).

`[y, freq] = frfun(Bf, jx, omg0, omgf, npts)` returns the computed frequency response in a three-dimensional matrix `y` and frequencies of computation in a vector `freq`. Here `omg0` and `omgf` are the lower and upper bounds of a frequency region for computation and `npts` is the number of points in the frequency region. Of the matrix `y`, the row vectors

`y(k, :, 1)` and `y(k, :, 2)`

contain the magnitude and phase of the steady-state displacement at the kth station, respectively. In other words, `plot(freq, y(k, :, 1))` and `plot(freq, y(k, :, 2))` will plot the magnitude and phase of the kth displacement versus frequency.

By default, `Bf` is a vector with the first element being one and the rest zeros, `jx` = 1, `omg0` = 0, `omgf` is set such that up to four natural frequencies are included in the frequency region for computation, and `npts` = 1,001. Any one of the input parameters `Bf`, `jx`, `omg0`, `omgf`, `npts` can be replaced by the placeholder `[]` for the default value.

The frequency response can also be plotted by

```
y = frfun(Bf)
plotfr(y, jx)
```

where `plotfr` is a function given in Window 3.10.

Window 3.10. Function `plotfr`

MATLAB Function: **`plotfr`**

Purpose:
To plot the frequency response of an M-DOF system obtained by function `frfun`.

Synopsis:

```
plotfr(y, jx)
plotfr(y, jx, opt)
```

Description:
`plotfr(y, jx)` plots the magnitude and phase of the steady-state displacement at the *jx*-th station versus frequency, where `y` is a three-dimensional matrix of frequency response data obtained by function `frfun` (see Window 3.9).

`plotfr(y, jx, opt)` plots the steady-state magnitude and phase with the following options:

`opt = 0`	plot both magnitude and phase
`opt = 1`	plot magnitude only
`opt = 2`	plot phase only

EXAMPLE 3.6

Consider the system

$$\begin{bmatrix} 7 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 30 \end{bmatrix} \{\ddot{x}(t)\} + \begin{bmatrix} 0.4 & -0.2 & 0 \\ -0.2 & 0.6 & -0.4 \\ 0 & -0.4 & 1 \end{bmatrix} \{\dot{x}(t)\} + \begin{bmatrix} 10 & -10 & 0 \\ -10 & 30 & -20 \\ 0 & -20 & 60 \end{bmatrix} \{x(t)\} = \begin{bmatrix} 0 \\ 0 \\ 3.75 \end{bmatrix} \cos \omega t$$

To obtain the steady-state displacement of the system at the second station, $x_2(t)$, type

```
>> M = diag([7 10 30]);
>> K = [10 -10 0; -10 30 -20; 0 -20 60];
>> C = [0.4 -0.2 0; -0.2 0.6 -0.4; 0 -0.4 1];
>> setmdof(M, K, C)
>> Bf = [0 0 3.75]';
>> y = frfun(Bf);
>> plotfr(y, 2)
```

which yields Fig. 11.3.7.

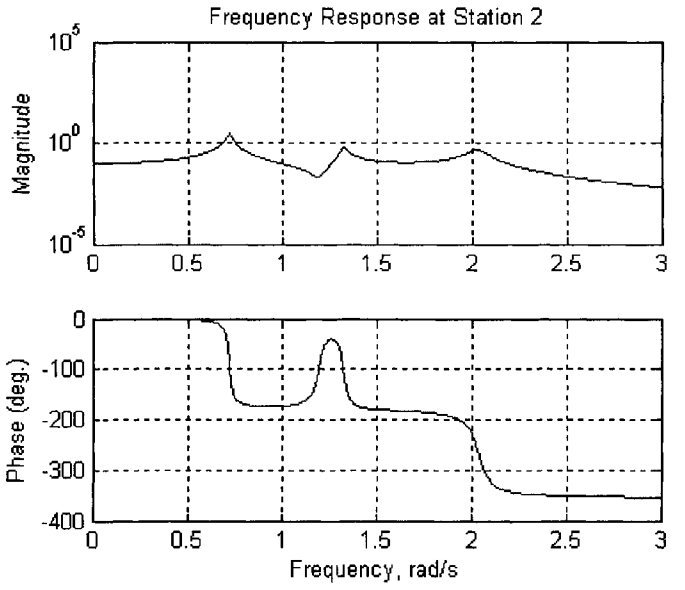


FIGURE 11.3.7

11.4 Dynamic Vibration Absorption

One important aspect in design of structures and machines is to suppress or reduce unwanted vibrations. Three commonly used techniques of vibration control are

- Introduction of damping
- Vibration isolation
- Dynamic vibration absorption

Damping, which is used to control vibration via energy dissipation, can be introduced through the installation of various types of dampers, treatment of viscoelastic damping layers, use of movable joints, and by other means. In vibration isolation, a resilient member or isolator is inserted to isolate the device or system from the source of vibration; see Chapter 10 for instance. In dynamic vibration absorption, a second mass-spring system (dynamic vibration absorber) is added to a given vibrating system (primary system), so as to minimize the vibration of the primary system. All these techniques are known as *passive vibration control methods*. For *active vibration control methods*, refer to Section 11.6.

This section is concerned with dynamic vibration absorption, which deals with an assembly of a primary system (a machine or device) and a dynamic vibration absorber. The primary system considered herein is a spring-mass system subject to a sinusoidal external force. The dynamic vibration absorber is another spring-mass system added to the primary system. The addition of the absorber turns a system with one degree of freedom to a system with two degrees of freedom. The parameters of the absorber are properly tuned so as to eliminate or suppress the steady-state vibration of the primary system. Vibration absorbers can be designed without or with damping.

11.4.1 Undamped Vibration Absorbers

Figure 11.4.1 shows a combination of a primary system (m, k) and a vibration absorber (m_a, k_a). The equations of motion for this two-degree-of-freedom system are

$$\begin{bmatrix} m & 0 \\ 0 & m_a \end{bmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{x}_a \end{pmatrix} + \begin{bmatrix} k + k_a & -k_a \\ -k_a & k_a \end{bmatrix} \begin{pmatrix} x \\ x_a \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} F_0 \sin \omega t \tag{11.64}$$

where x is the displacement of the primary mass m, and x_a is the displacement of the absorber mass m_a. The steady-state response of the system, under the sinusoidal excitation $F_0 \sin \omega t$, is

$$\begin{pmatrix} x \\ x_a \end{pmatrix} = \begin{pmatrix} X \\ X_a \end{pmatrix} \sin \omega t \tag{11.65}$$

where the displacement amplitudes by Eq. (11.58) are obtained as

$$\begin{pmatrix} X \\ X_a \end{pmatrix} = \frac{F_0}{(k + k_a - m\omega^2)(k_a - m_a\omega^2) - k_a^2} \begin{pmatrix} k_a - m_a\omega^2 \\ k_a \end{pmatrix} \tag{11.66}$$

If $k_a = \omega^2 m_a$, the steady-state motion of the primary mass is zero ($x = 0$), and the vibration of the absorber mass becomes

$$x_a = -\frac{F_0}{k_a} \sin \omega t \tag{11.67}$$

This indicates that the absorber parameters can be tuned to suppress the steady-state vibration of the primary system.

For the convenience of discussion, define the following parameters:

$r = \dfrac{\omega}{\omega_p}$ forced frequency ratio

$\mu = \dfrac{m_a}{m}$ mass ratio

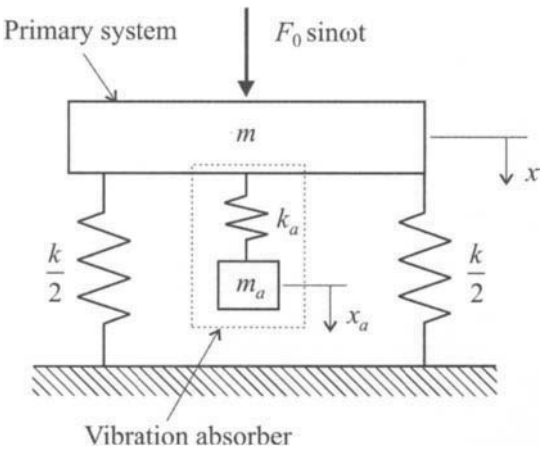


FIGURE 11.4.1 Undamped vibration absorber.

$\omega_p = \sqrt{\dfrac{k}{m}}$ the natural frequency of the primary system

$\omega_a = \sqrt{\dfrac{k_a}{m_a}}$ the natural frequency of the absorber

$\alpha = \dfrac{\omega_a}{\omega_p}$ frequency ratio (ratio of natural frequencies)

With the above-defined parameters, the displacement amplitudes given in Eq. (11.66) are rewritten in the nondimensional form

$$\begin{pmatrix} \dfrac{X}{F_0/k} \\ \dfrac{X_a}{F_0/k} \end{pmatrix} = \frac{1}{\left(1+\mu\alpha^2-r^2\right)\left(\alpha^2-r^2\right)-\mu\alpha^4} \begin{pmatrix} \alpha^2-r^2 \\ \alpha^2 \end{pmatrix} \tag{11.68}$$

Natural Frequencies The characteristic equation of the combined system, by Eq. (11.68), is

$$\left(1+\mu\alpha^2-r^2\right)\left(\alpha^2-r^2\right)-\mu\alpha^4 = 0 \tag{11.69}$$

The positive roots of the above equation are

$$\begin{matrix} r_1 \\ r_2 \end{matrix} = \frac{1}{\sqrt{2}}\sqrt{1+(1+\mu)\alpha^2 \mp \sqrt{\left(1+(1+\mu)\alpha^2\right)^2-4\alpha^2}} \tag{11.70}$$

by which the two natural frequencies of the combined system are given by

$$\Omega_j = r_j\,\omega_p, \quad j = 1,2 \tag{11.71}$$

It can be shown that the natural frequencies of the combined system are always separated by the natural frequency of the primary system; namely,

$$\Omega_1 < \omega_p < \Omega_2 \tag{11.72}$$

Operating Range In design of a vibration absorber, the natural frequency of the absorber is selected such that the steady-state amplitude of the primary mass is zero at the frequency ω of the driving force. This requires that $\omega_a = \omega$ or $\alpha = r$. In practice, the driving frequency ω may drift from ω_a, rendering large vibration or even resonance. If the driving frequency ω is within a neighborhood of ω_a such that the normalized amplitude of the primary mass satisfies the condition

$$\left|\frac{X}{F_0/k}\right| = \left|\frac{\alpha^2-r^2}{\left(1-r^2\right)\left(\alpha^2-r^2\right)-\mu\alpha^2 r^2}\right| \le 1 \tag{11.73}$$

the absorber still provides some protection to the primary system. This defines an operating range for the absorber

$$r_L \le r \le r_R \tag{11.74a}$$

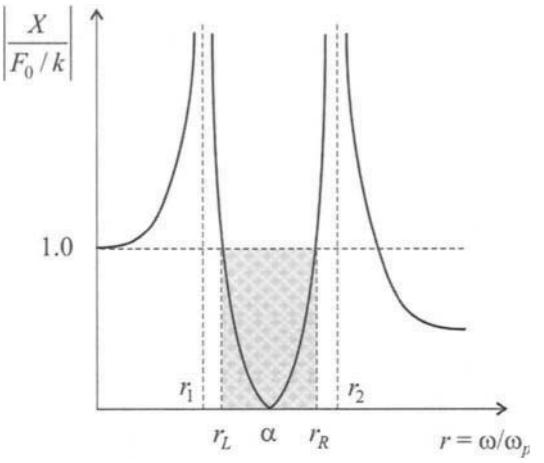


FIGURE 11.4.2 The operating range of an undamped vibration absorber.

or

$$r_L \cdot \omega_p \le \omega \le r_R \cdot \omega_p \tag{11.74b}$$

in which the shift in driving frequency is tolerable; see the shaded area in Fig. 11.4.2. By letting the equality of Eq. (11.73) hold, the lower and upper bounds of the operating range are determined as

$$\begin{aligned} r_L &= \tfrac{1}{\sqrt{2}}\sqrt{2+(1+\mu)\alpha^2 - \sqrt{\left(2+(1+\mu)\alpha^2\right)^2 - 8\alpha^2}} \\ r_R &= \alpha\sqrt{1+\mu} \end{aligned} \tag{11.75}$$

The bandwidth of the operating range ($\Delta r = r_R - r_L$) increases as the mass ratio μ increases.

Design Criteria The primary system without the added absorber is in resonant motion when the driving frequency is close to its natural frequency, i.e., $\omega^2 = k/m$. So, of special interest is to suppress the vibration of the primary system near its resonance. To this end, the following conditions are commonly used in an absorber design:

$$\text{(C1)} \qquad \frac{k_a}{m_a} = \frac{k}{m} \tag{11.76a}$$

$$\text{(C2)} \qquad \mu_{\min} \le \mu \le \mu_{\max} \tag{11.76b}$$

Condition (C1) requires that the natural frequency of the absorber coincides with that of the primary system; namely, $\alpha = 1$. This results in suppressed steady-state motion of the primary system for $\omega \approx \omega_p$. Condition (C2) defines a range of mass ratio m_a/m for proper operation of the absorber. As a rule of thumb, the value of μ normally falls between 0.05 and 0.25. Conditions (C1) and (C2) also imply that $\mu_{\min} k \le k_a \le \mu_{\max} k$.

The Toolbox of this chapter has a function `absorb` for characterizing an undamped vibration absorber; see Window 4.1.

Window 4.1. Function absorb

MATLAB Function: **absorb**

Purpose:
To characterize dynamic behaviors of a combined system consisting of a primary system and an undamped vibration absorber.

Synopsis:

```
absorb(Mp, Kp, mass_ratio, freq_ratio)
```

Description:
`absorb(Mp, Kp, mass_ratio, freq_ratio)` determines the natural frequencies of the combined system and the parameters and operating range of the vibration absorber, where `Mp` and `Kp` are the mass and stiffness of the primary system, `mass_ratio` and `freq_ratio` are the mass ratio and the ratio of natural frequencies, respectively. By default, `mass_ratio = 0.25` and `freq_ratio = 1`. In addition, function `absorb` also plots the following four figures: the steady-state displacement amplitude of the primary mass versus frequency, the steady-state displacement amplitude of the absorber mass versus frequency, the operating range of the absorber versus mass ratio, and the natural frequency loci of the combined system versus mass ratio.

EXAMPLE 4.1

A vibrating system of mass $m = 200$ kg and stiffness $k = 1.5 \times 10^6$ N/m is subject to a sinusoidal force of amplitude $F_0 = 2500\ N$ and frequency $\omega = 105$ rad/s. Design a dynamic vibration absorber such that the vibration of the primary mass ceases and the steady-state amplitude of the absorber mass is less than 0.01 m. Also, determine the operating range of the vibration absorber and the natural frequencies of the combined system.

To eliminate the steady-state vibration of the primary mass, the natural frequency of the absorber is tuned to coincide with the excitation frequency, namely, $\omega_a = 105$ rad/s. Because the natural frequency of the primary system is $\omega_p = \sqrt{k/m} = 86.6025$ rad/s, the frequency ratio is $\alpha = \omega_a/\omega_p = 1.2124$. By Eq. (11.67), the steady-state absorber amplitude needs to satisfy

$$|X_a| = \frac{F_0}{k_a} < 0.01 \text{ m}$$

or

$$k_a > \frac{2500\ N}{0.01 \text{ m}} = 2.5 \times 10^5 \text{ N/m}$$

which implies that the mass ratio

$$\frac{m_a}{m} = \frac{1}{\alpha^2}\frac{k_a}{k} > \frac{1}{(1.2124)^2}\frac{2.5 \times 10^5}{1.5 \times 10^6} = 0.1134$$

Choose $\mu = 0.15$. The MATLAB command

```
>> absorb(200,1.5e6, 0.15, 1.2124)
```

yields the parameters and operating range of the absorber and the natural frequencies of the combined system as below.

```
Design Result
-------------
 Absorber Parameters:
   Ma (mass) = 30
   Ka (stiffness) = 330730.596
   natural frequency wa = 104.9969 rad/s

 Operating Range (w1 <= Omega <= w2):
   w1 = 93.4079 rad/s
   w2 = 112.5967 rad/s
   Bandwidth = w2 - w1 = 19.1888 rad/s

 Natural Frequencies of the Combined System:
   Omega_1 = 75.6177 rad/s
   Omega_2 = 120.2496 rad/s
```

The execution of function absorber also yields the steady-state amplitudes of the primary mass and absorber plotted in Figs. 11.4.3 and 11.4.4.

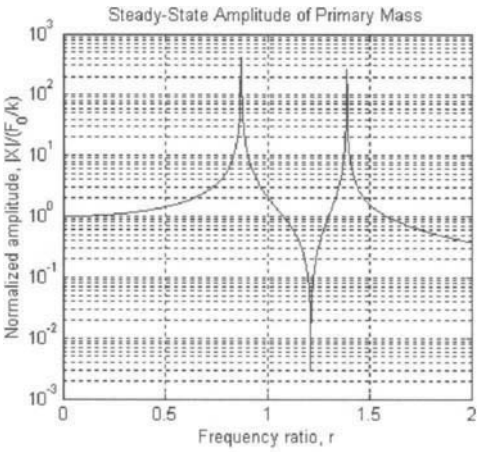


FIGURE 11.4.3

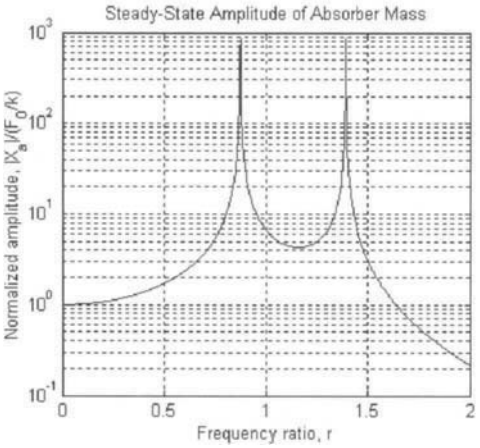


FIGURE 11.4.4

11.4.2 Damped Vibration Absorbers

Damping can be introduced in an absorber design; see Fig. 11.4.5, where c_a is a viscous damping coefficient. The purpose here is to widen the bandwidth of operating range and to avoid resonance. The trade-off of adding damping, however, is that the steady-state displacement of the primary mass cannot be completely suppressed.

For the combined system shown in Fig. 11.4.5, the equations of motion are

$$\begin{bmatrix} m & 0 \\ 0 & m_a \end{bmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{x}_a \end{pmatrix} + \begin{bmatrix} c_a & -c_a \\ -c_a & c_a \end{bmatrix} \begin{pmatrix} \dot{x} \\ \dot{x}_a \end{pmatrix} + \begin{bmatrix} k + k_a & -k_a \\ -k_a & k_a \end{bmatrix} \begin{pmatrix} x \\ x_a \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} F_0 \sin \omega t \tag{11.77}$$

Following Section 11.3.6, the steady-state motion of the combined system is determined as

$$\begin{aligned} x &= X \sin(\omega t + \phi) \\ x_a &= X_a \sin(\omega t + \phi_a) \end{aligned} \tag{11.78}$$

where the magnitudes

$$\begin{aligned} X &= \frac{F_0}{k} \sqrt{\frac{\left(r^2 - \alpha^2\right)^2 + 4\xi^2 \alpha^2 r^2}{\Delta(r)}} \\ X_a &= \frac{F_0}{k} \sqrt{\frac{\alpha^4 + 4\xi^2 \alpha^2 r^2}{\Delta(r)}} \end{aligned} \tag{11.79}$$

with

$$\Delta(r) = \left[\left(r^2 - 1\right)\left(r^2 - \alpha^2\right) - \mu \alpha^2 r^2\right]^2 + 4\xi^2 \alpha^2 r^2 \left[1 - (1 + \mu) r^2\right]^2 \tag{11.80}$$

and the phase angles

$$\phi = \tan^{-1}\left(\frac{2\xi \alpha r}{\alpha^2 - r^2}\right), \quad \phi_a = \tan^{-1}\left(\frac{2\xi r}{\alpha}\right). \tag{11.81}$$

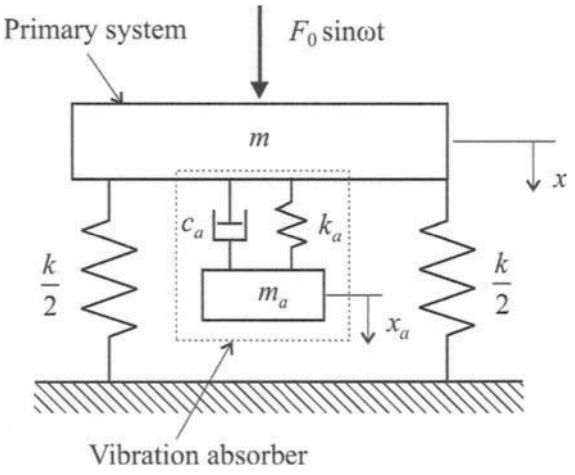


FIGURE 11.4.5 Damped vibration absorber.

The parameters α, μ, and r have been introduced in Section 11.4.1. The parameter ξ is the damping ratio of the absorber defined by

$$\xi = \frac{c_a}{2m_a\omega_a} = \frac{c_a}{2\sqrt{m_a k_a}} \tag{11.82}$$

Fixed Points of Amplitude Curves Given mass ratio μ and frequency ratio α, the normalized steady-state amplitude $X/(F_0/k)$ of the primary mass can be plotted against the forced frequency ratio r, for different values of damping ratio ξ; see Fig. 11.4.6 for $\mu = 0.1$, $\alpha = 1.1$, and $\xi = 0.01$, 0.1, 0.2, and 0.3. Note that all the curves pass through two points A and B, independent of the value of ξ. These points are known as *fixed points*. The locations of the fixed points in the graph are given by

$$\begin{matrix} r_A \\ r_B \end{matrix} = \sqrt{\frac{1 + (1+\mu)\alpha^2 \mp \sqrt{1 + (1+\mu)^2\alpha^4 - 2\alpha^2}}{2+\mu}} \tag{11.83}$$

and the values of $X/(F_0/k)$ at the points are determined by

$$\frac{X}{F_0/k} = \frac{1}{|1 - (1+\mu)r^2|} \quad \text{for } r = r_A,\ r_B \tag{11.84}$$

Furthermore, the value of $X/(F_0/k)$ is the same at r_A and r_B if the frequency ratio

$$\alpha = \frac{1}{1+\mu} \tag{11.85}$$

The existence of the fixed points provides an approach to optimal tuning of the damped vibration absorber, as to be presented in sequel.

Optimal Absorber Design The purpose of optimal tuning of a damped vibration absorber is to minimize the steady-state amplitude of the primary mass over the entire range of driving frequency. According to Fig. 11.4.6, the minimum amplitude can be achieved

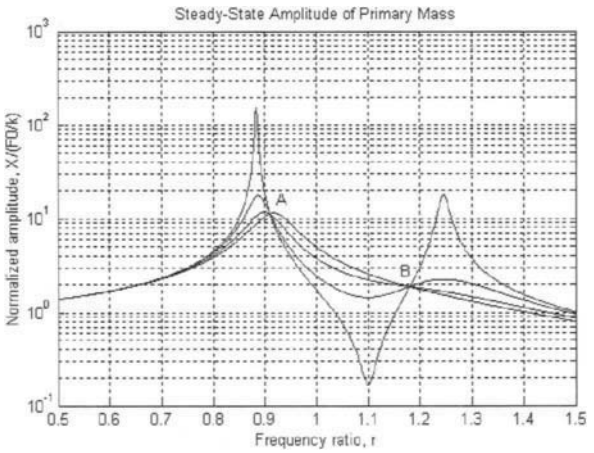


FIGURE 11.4.6 Normalized steady-state amplitude of the primary mass.

when the ordinates of the fixed points A, B are the same. This leads to an absorber tuning schedule as follows:

Step 1. Select a desired mass ratio μ.

Step 2. Determine the frequency ratio by Eq. (11.85) such that the values of $X/(F_0/k)$ at the fixed points are the same. Under this condition, the normalized amplitude of the primary mass is a function of r, ξ, and μ

$$\frac{X}{F_0/k} = G(r, \xi, \mu) \tag{11.86}$$

Note that the value of μ has been selected in Step 1.

Step 3. Determine the optimal damping ratio ξ_{opt} that renders the minimum amplitude of the primary mass. Mathematically, this is a minimum-maximum optimization problem

$$\min_{\xi} \left\{ \max_{r} G(r, \xi, \mu) \right\} \tag{11.87}$$

which can be solved by function `absorbopt` from the Toolbox of this chapter (Window 4.4). On the other hand, an approximate estimation of the optimal damping is

$$\xi_{\text{opt}} \approx \sqrt{\frac{3\mu}{8(1+\mu)}} \tag{11.88}$$

which is obtained based on the assumption that the maximum of $G(r, \xi, \mu)$ is at one of the fixed points A, B. Although the actual maximum is not at any fixed point, formula (11.88) is accurate enough for a relatively small mass ratio, say $\mu < 0.5$.

Note 1. The larger the mass ratio μ, the smaller the maximum amplitude of the primary mass is.

Note 2. The damping ratio considered here is defined with the natural frequency of the absorber; see Eq. (11.82). The damping ratio can also be defined with the natural frequency of the primary system, i.e., $\xi = c_a/(2m_a\omega_p)$. In that case, the approximate optimal damping ratio is given by (Reference 5)

$$\xi_{\text{opt}} \approx \sqrt{\frac{3\mu}{8(1+\mu)^3}}.$$

The Toolbox of this chapter provides functions `absorbdmp`, `absorbmsratio`, and `absorbopt` for plotting the steady-state response of the combined system and for optimal absorber design; see Windows 4.2 to 4.4.

Window 4.2. Function absorbdmp

MATLAB Function: **absorbdmp**

Purpose:
To plot the steady-state amplitudes of a combined system consisting of a primary system and a damped vibration absorber.

Window 4.2. Function absorbdmp (*Continued*)

Synopsis:

```
absorbdmp(mass_ratio, freq_ratio, dmp_ratios)
[r, xp, xa] = absorbdmp(mass_ratio, freq_ratio, dmp_ratios, plot_opt)
```

Description:

`absorbdmp(mass_ratio, freq_ratio, dmp_ratios)` plots the steady-state amplitudes of the primary mass and absorber mass against forced frequency ratio, where `mass_ratio` and `freq_ratio` are the mass ratio and the frequency ratio, respectively, and `dmp_ratios` is a vector of damping ratios. By default, `mass_ratio = 0.25`, `freq_ratio = 1`, and `dmp_ratios = [0 0.05 0.1 0.15 0.2 0.3 0.5 0.7 1]`. The `mass_ratio`, `freq_ratio`, and `dmp_ratios` can be replaced by `[]` as a placeholder for default value.

`[r, xp, xa] = absorbdmp(mass_ratio, freq_ratio, dmp_ratios, plot_opt)` returns the normalized frequencies in a vector `r`, the amplitude of the primary mass in a matrix `xp`, and the amplitude of the absorber mass in a matrix `xa`. For instance, `plot(r, xp(k,:))` plots the amplitude of the primary mass for the damping ratio `dmp_ratios(k)`. Also, the input argument `plot_opt` assigns the following plotting options:

`plot_opt = 0` or `[]`	plot with logarithmic scale for the amplitude (y-axis)
`plot_opt = 1`	plot with regular scales for x- and y- axes

Window 4.3. Function absorbmsratio

MATLAB Function: **absorbmsratio**

Purpose:

To obtain a range of mass ratio for vibration absorber design.

Synopsis:

```
absorbmsratio(Mp, Kp, F0, Xp_max)
mu_cr = absorbmsratio(Mp, Kp, F0, Xp_max)
```

Description:

`absorbmsratio(Mp, Kp, F0, Xp_max)` determines a lower bound μ_{cr} of mass ratio for a vibration absorber to be optimally tuned, where `Mp` and `Kp` are the mass and stiffness of the primary system, `F0` is the magnitude of the driving force, and `Xp_max` is a specified upper bound of the amplitude of the primary system. The μ_{cr} is such that the amplitude of the primary mass $X <$ `Xp_max` for mass ratio $\mu > \mu_{cr}$.

`mu_cr = absorbmsratio(Mp, Kp, F0, Xp_max)` returns the value of the lower bound μ_{cr} of mass ratio in `mu_cr`.

Window 4.4. Function absorbopt

MATLAB Function: **absorbopt**

Purpose:
To optimally tune a damped vibration absorber for a given primary system.

Synopsis:

```
absorbopt(Mp, Kp, mass_ratio, plot_opt)
[d_opt, freq_opt, xmax, ma, ca, ka] = absorbopt(Mp, Kp,
mass_ratio, plot_opt)
```

Description:
`absorbopt(Mp, Kp, mass_ratio, plot_opt)` determines the optimal parameters of the vibration absorber that minimizes the steady-state amplitude of the primary mass and plots the steady-state amplitudes of the combined system with such tuned absorber, where `Mp` and `Kp` are the mass and stiffness of the primary system, `mass_ratio` is a given mass ratio, and `plot_opt` is a plotting option number as described in Window 4.2. By default, `mass_ratio = 0.25` and `plot_opt = 0`.

`[d_opt, freq_opt, xmax, ma, ca, ka] = absorbopt(Mp, Kp, mass_ratio, plot_opt)` returns the optimal damping ratio `d_opt`, the optimal frequency ratio `freq_opt`, the maximum normalized amplitude $X/(F_0/k)$ of the primary mass in `xmax`, and the optimally tuned parameters (`ma`, `ca`, `ka`) of the vibration absorber.

EXAMPLE 4.2

A combined system (primary mass + absorber) has mass ratio $\mu = 0.1$ and frequency ratio $\alpha = 1.1$. For absorber damping ratio $\xi = 0.01, 0.1, 0.2$, and 0.3,

```
>> absorbdmp(0.1, 1.1, [0.01 0.1 0.2 0.3])
```

produces the curves of the primary mass amplitude versus the forced frequency ratio in Fig. 11.4.6, and determines the fixed points of the curves as follows

```
Fixed points of frequency response curves of the primary mass:
   r1 = 0.90976, r2 = 1.18
Normalized amplitude of the primary mass at fixed points:
   X/(F0/k) at r1 = 11.1632
   X/(F0/k) at r2 = 1.8812
```

EXAMPLE 4.3

A machine of mass $m = 340$ kg and natural frequency $\omega_p = 145$ rad/s is subject to a harmonic excitation of magnitude $F_0 = 12{,}000$ N and driving frequency varying in a wide range. Design a damped vibration absorber such that the amplitude of the machine is less than 0.006 m.

The stiffness of the machine is $k = m\omega_p^2 = 340 \times (145)^2 = 7{,}148{,}500$ N/m. By

```
>> absorbmsratio(340, 7148500, 12000, 0.006)
```

it is found that the mass ratio should be greater than 0.17, in order to meet the amplitude condition $X < 0.006$ m. Let $\mu = 0.18$. The command

```
>> absorbopt(340, 7148500, 0.18)
```

yields the parameters of the optimal absorber as follows:

```
Result on Optimal Absorber Design
---------------------------------
   Optimal frequency ratio = 0.84746
   Optimal damping ratio zeta_opt = 0.24019
   Approx optimal damping ratio, sqrt(3*mu/8/(1+mu)) = 0.23917
   Ma (mass) = 61.2
   Ca (damping coeff) = 3612.6097
   Ka (stiffness) = 924109.4513
   Natural frequency wa = 122.8814
   Maximum normalized amplitude of the primary mass:
      X/(F0/k) = 3.4887, at point r = 1.0559
```

and the amplitude-frequency curves at the optimal damping ratio and two other comparing damping ratios as plotted in Fig. 11.4.7. The maximum amplitude of the machine under the so designed absorber is

$$X_{\max} = 3.4887\frac{F_0}{k} = 3.4887\frac{12{,}000}{7{,}148{,}500} = 0.0059\,\text{m}$$

which is below the desired 0.006 m.

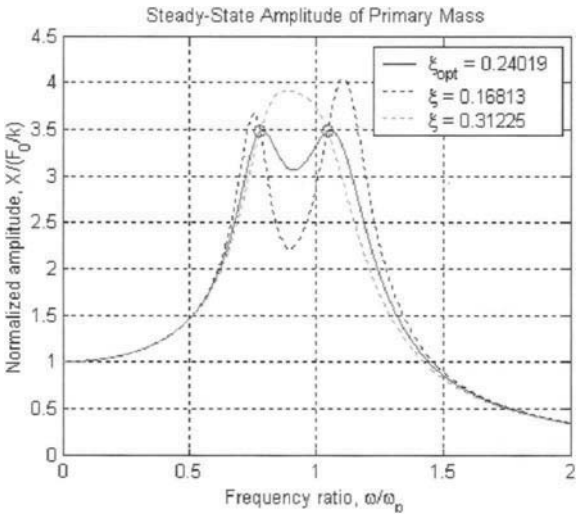


FIGURE 11.4.7

11.5 Transfer Function Formulation

11.5.1 Transfer Function and Green's Function

In dynamic analysis and feedback control of M-DOF mechanical systems, transfer functions and Green's functions are commonly used (References 8 and 9). To define these functions, consider a system of n degrees of freedom

$$\begin{gathered}[M]\{\ddot{x}(t)\}+[C]\{\dot{x}(t)\}+[K]\{x(t)\}=\left[B_f\right]\{F(t)\}, \quad t\geq 0 \\ \{x(0)\}=\{x_0\}, \quad \{\dot{x}(0)\}=\{v_0\}\end{gathered} \tag{11.89}$$

where $\{F(t)\}=\left(f_1(t) \quad f_2(t) \quad \cdots \quad f_{n_f}(t)\right)^T$ is a vector of forcing functions, and $\left[B_f\right]$ is an $n \times n_f$ force distribution matrix. Laplace transform of Eq. (11.89) with respect to time gives

$$\left(s^2[M]+s[C]+[K]\right)\{X(s)\}=[M]\left(s\{x_0\}+\{v_0\}\right)+[C]\{x_0\}+\left[B_f\right]\{F(s)\} \tag{11.90}$$

where $\{X(s)\}$ and $\{F(s)\}$ are the Laplace transforms of $\{x(t)\}$ and $\{F(t)\}$, respectively. It follows that

$$\{X(s)\}=H(s)\left(s[M]\{x_0\}+[M]\{v_0\}+[C]\{x_0\}\right)+H(s)\left[B_f\right]\{F(s)\} \tag{11.91}$$

where the $n \times n$ matrix $H(s)$ is the transfer function of the system, and it has the form

$$H(s)=\left(s^2[M]+s[C]+[K]\right)^{-1} \tag{11.92}$$

Taking inverse Laplace transform of Eq. (11.91) gives the time response of the system

$$\{x(t)\}=\frac{d}{dt}G(t)[M]\{x_0\}+G(t)\left([M]\{v_0\}+[C]\{x_0\}\right)+\int_0^\tau G(t-\tau)\left[B_f\right]\{F(\tau)\}\,d\tau \tag{11.93}$$

The $n \times n$ matrix $G(t)$ is the Green's function of the system, and it is given by

$$G(t)=\mathcal{L}^{-1}[H(s)]=\mathcal{L}^{-1}\left[\left(s^2[M]+s[C]+[K]\right)^{-1}\right] \tag{11.94}$$

where $\mathcal{L}^{-1}$ is the inverse Laplace transform operator. The $G(t)$ is also called the *impulse response* of the system because it is the solution of

$$\begin{gathered}[M]\left\{\ddot{G}(t)\right\}+[C]\left\{\dot{G}(t)\right\}+[K]\{G(t)\}=[I]\,\delta(t), \quad t\geq 0 \\ \{G(0)\}=0, \quad \left\{\dot{G}(0)\right\}=0\end{gathered} \tag{11.95}$$

where $\delta(t)$ is the Dirac delta function and $[I]$ is the identity matrix. Equation (11.95) can be used to plot the Green's function by the MATLAB functions `forcedMD`, `forcedLT`, and `forcedRK`; see Example 5.1. Equation (11.93) is called the *Green's function formula.*

Now, define another transfer function

$$H_f(s)=\left(s^2[M]+s[C]+[K]\right)^{-1}\left[B_f\right] \tag{11.96}$$

It is seen that $H_f(s) = H(s)$ when $[B_f]$ is an identity matrix. According to Eq. (11.91), the system response in the s-domain (under zero initial conditions) is

$$\{X(s)\} = H_f(s)\{F(s)\} \tag{11.97}$$

Write $H_f(s) = [h_{ij}(s)]$. The entry $h_{ij}(s) = X_i(s)/F_j(s)$, which is the transfer function from the jth external load to the ith displacement. Additionally, define another Green's function by

$$G_f(t) = \mathcal{L}^{-1}[H_f(s)] \tag{11.98}$$

When $[B_f]$ is an identity matrix, $G_f(t) = G(t)$.

Solution by the Toolbox The Toolbox of this chapter provides a function `getTF` for obtaining transfer function $H_f(s)$, and function `getGF` for obtaining Green's function $G_f(t)$; see Windows 5.1 and 5.2.

Window 5.1. Function getTF

MATLAB Function: **getTF**

Purpose:
To obtain transfer function $H_f(s)$ of an M-DOF system.

Synopsis:

```
getTF(Bf)
[num, den] = getTF(Bf)
```

Description:
`getTF(Bf)` displays the mathematical expressions of transfer function $H_f(s)$ for a given force distribution matrix `Bf`. Also, `getTF` displays the transfer function $H(s)$, i.e., $H_f(s)$ with `Bf` being an identity matrix.

`[num, den] = getTF(Bf)` returns the numerators of $H_f(s)$ in a three-dimensional matrix `num` and the denominator of $H_f(s)$ in a vector `den`. `[num, den] = getTF` returns the numerators and denominator of $H(s)$.

In function `getTF`, the transfer function $H_f(s)$ is expressed as

$$H_f(s) = \frac{1}{\Delta(s)}[A(s)] = \frac{1}{\Delta(s)}\begin{bmatrix} a_{11}(s) & \cdots & a_{1n_f}(s) \\ \vdots & & \vdots \\ a_{n1}(s) & \cdots & a_{nn_f}(s) \end{bmatrix} \tag{11.99}$$

where $\Delta(s) = \det\left(s^2[M] + s[C] + [K]\right)$, $a_{ij}(s)$ are polynomials of s, `n` is the number of degrees of freedom of the system, and n_f is the number of the external forces (see Eq. (11.89)). In Window 5.1, the vector `den` contains the coefficients of the characteristic polynomial $\Delta(s)$; while `num` is an $n \times (2n+1) \times n_f$ matrix with its row vectors describing the entries of $[A(s)]$, namely, the vector `num(i, :, j)` contains the coefficients of $a_{ij}(s)$.

Window 5.2. Function getGF

MATLAB Function: **getGF**

Purpose:
To obtain Green's function of an M-DOF system.

Synopsis:

```
getGF(Bf)
y = getGF(Bf)
```

Description:
`getGF(Bf)` displays the mathematical expressions of the Green's function $G_f(t)$ for a given force distribution matrix `Bf`. `getGF` displays the Green's function $G(t)$, i.e., $G_f(t)$ with `Bf` being an identity matrix.

`y = getGF(Bf)` returns the coefficients of the mathematical expressions of $G_f(t) = [g_{ij}(t)]$ in a four-dimensional matrix `y`. The coefficients of the function $g_{ij}(t)$ are contained in the submatrix `y(:, :, i, j)`, the row vectors of which have the format specified in Table 11.3.1. For instance, the submatrix

$$\texttt{y(:, :, 1, 2)} = \begin{bmatrix} 5 & 0.1 & -0.2 & -0.25 & 4.1 & 0 \\ 5 & -0.02 & 0.09 & -0.57 & 12.8 & 0 \end{bmatrix}$$

indicates that

$$g_{12}(t) = e^{-0.25t}(0.1\cos 4.1t - 0.2\sin 4.1t) - e^{-0.57t}(0.02\cos 12.8t - 0.09\sin 12.8t).$$

EXAMPLE 5.1

Consider the spring-mass-damper system in Fig. 11.1.1, with $m_1 = m_2 = 1$, $c_1 = 1$, $c_2 = 2$, $k_1 = 3$, and $k_2 = 4$. The system parameter matrices are

$$[M] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad [C] = \begin{bmatrix} 3 & -2 \\ -2 & 2 \end{bmatrix}, \quad [K] = \begin{bmatrix} 7 & -4 \\ -4 & 4 \end{bmatrix}$$

By the commands

```
>> m=diag([1 1]); c=[3 -2; -2 2]; k = [7 -4; -4 4];
>> setmdof(m, k, c)
>> getTF
```

the transfer function of the system is obtained as follows

$$H(s) = \begin{bmatrix} s^2+3s+7 & -2s-4 \\ -2s-4 & s^2+2s+4 \end{bmatrix}^{-1} = \frac{1}{\Delta(s)} \begin{bmatrix} s^2+2s+4 & 2s+4 \\ 2s+4 & s^2+3s+7 \end{bmatrix}$$

where $\Delta(s) = s^4 + 5s^3 + 13s^2 + 10s + 12$. Also,

```
>> getGF
```

gives the Green's function of the system as follows

$$G(t) = \mathcal{L}^{-1}[H(s)] = \begin{bmatrix} g_{11}(t) & g_{12}(t) \\ g_{12}(t) & g_{22}(t) \end{bmatrix}$$

where

$$g_{11}(t) = e^{-\sigma_1 t}(0.0234\cos\omega_1 t + 0.3076\sin\omega_1 t) + e^{-\sigma_2 t}(-0.0234\cos\omega_2 t + 0.2897\sin\omega_2 t)$$

$$g_{12}(t) = e^{-\sigma_1 t}(0.0073\cos\omega_1 t + 0.4360\sin\omega_1 t) - e^{-\sigma_2 t}(0.0073\cos\omega_2 t + 0.2298\sin\omega_2 t)$$

$$g_{22}(t) = e^{-\sigma_1 t}(-0.0262\cos\omega_1 t + 0.6158\sin\omega_1 t) + e^{-\sigma_2 t}(0.0262\cos\omega_2 t + 0.1799\sin\omega_2 t)$$

and

$$\sigma_1 = 0.22546, \quad \omega_1 = 1.0884, \quad \sigma_2 = 2.2745, \quad \omega_2 = 2.1307.$$

The above Green's function can be plotted based on Eq. (11.95). For instance, if function `forcedRK` is used, the following commands

```
>> load = [0 1]; % unite impulse force
>> Bf1 = [1; 0]; % for g11 and g12
>> [g1, t] = forcedRK(load, Bf1);
>> Bf2 = [0; 1]; % for g12 and g22
>> g2 = forcedRK(load, Bf2);
>> plot(t, g1(1,:), t, g1(2,:), t, g2(2,:))
>> grid
>> xlabel('time, t')
```

plot the Green's function in Fig. 11.5.1. Besides `forcedRK`, function `forcedMD`, or `forcedLT` can also be used to plot the Green's function.

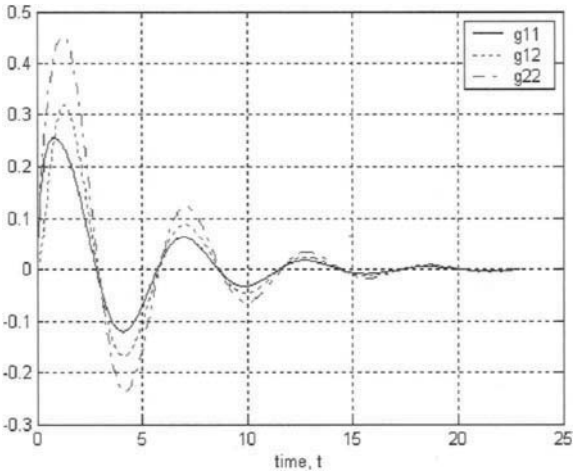


FIGURE 11.5.1

11.5.2 Open-Loop Transfer Function

Consider an M-DOF mechanical system controlled via a pair of sensor and actuator, which can be described as

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \{B_c\}u(t) \tag{11.100a}$$

$$y(t) = \{C_s\}^T\{x(t)\} \tag{11.100b}$$

where $u(t)$ is the control force applied to the mechanical system by the actuator, $y(t)$ is the output of the sensor on the system response, and $\{B_c\}$, $\{C_s\}$ are constant vectors specifying the distribution of actuation and sensing, respectively. The transfer function from u to y is

$$G_p(s) = \frac{Y(s)}{U(s)} = \{C_s\}^T H(s)\{B_c\} \tag{11.101}$$

where the transfer function $H(s)$ has been defined in Eq. (11.92). The $G_p(s)$ is known as the *open-loop transfer function*. See Section 11.6 for closed-loop formulation for feedback control of M-DOF systems.

Modal Expansion Assume that the mechanical system in consideration is undamped or proportionally damped. (See Section 11.2.1 for the definition of proportional damping.) Following Section 11.3.1 (modal analysis), $H(s)$ is expressed as

$$H(s) = \sum_{k=1}^{n} \frac{1}{s^2 + c_k s + \omega_k^2}\{u_k\}\{u_k\}^T \tag{11.102}$$

where ω_k are the undamped (natural) frequencies, $\{u_k\}$ are the orthonormal eigenvectors defined by Eqs. (11.6), and $c_k = \{u_k\}^T[C]\{u_k\}$. It follows that

$$G_p(s) = \frac{Y(s)}{F(s)} = \sum_{k=1}^{n} \frac{a_k b_k}{s^2 + c_k s + \omega_k^2} \tag{11.103}$$

with

$$a_k = \{C_s\}^T\{u_k\}, \quad b_k = \{u_k\}^T\{B_c\} \tag{11.104}$$

Actuator and Sensor Locations The form of vectors $\{B_c\}$ and $\{C_s\}$ depends on the model of the mechanical system and sensor and actuation locations. For a spring-mass-damper system or a rigid-body system with the sensor located at the jth station (corresponding to x_j) and the actuator located at the kth station (corresponding to x_k), $\{C_s\} = \{e_j\}$ and $\{B_c\} = \{e_k\}$, where $\{e_j\}$ is an n-vector with the jth element being one and the rest zeros, and $\{e_k\}$ is similarly defined. If Eq. (11.100a) is obtained from discretization of a flexible system or structure (say by a finite element method), vectors $\{B_c\}$ and $\{C_s\}$ are usually full with nonzero elements.

If $\{C_s\} = \{B_c\}$, the sensor and actuator are *colocated* and the control system is a colocated control system. If $\{C_s\} \neq \{B_c\}$, the sensor and actuator are *noncolocated*, and the control system is a noncolocated control system.

If the coefficient a_k in Eq. (11.103) is zero, the sensor is located at a nodal point of the kth mode shape of the mechanical system, and hence cannot pick up the information on the vibration of the system in that mode. Likewise, if the coefficient b_k in Eq. (11.103) is zero, the actuator is located at a nodal point of the kth mode shape, and hence cannot affect the vibration of the system in that mode.

Poles and Zeros The poles of transfer function $G_p(s)$, by Eq. (11.103), are determined by $s^2 + c_k s + \omega_k^2 = 0$. The zeros of the transfer function depend on the sensor and actuator locations. For a colocated undamped control system ($\{C_s\} = \{B_c\}$, $c_k = 0$ for all k), the open-loop transfer function becomes

$$G_p(s) = \sum_{k=1}^{n} \frac{a_k^2}{s^2 + \omega_k^2} \tag{11.105}$$

which has $2n$ imaginary poles $\pm j\omega_k$, $k = 1, 2, \ldots, n$, $j = \sqrt{-1}$, and up to $2n-2$ imaginary zeros $\pm j\alpha_k$, $k = 1, 2, \ldots, n-1$. Furthermore, if $a_k \neq 0$ for all k, the poles and zeros alternatively located on the imaginary axis; that is,

$$\omega_1 < \alpha_1 < \omega_2 < \cdots < \alpha_{n-1} < \omega_n \tag{11.106}$$

This interesting *pole-zero alternation* or interlacing can be easily proven via plotting

$$G_p(j\omega) = \sum_{k=1}^{n} \frac{a_k^2}{-\omega^2 + \omega_k^2} \tag{11.107}$$

versus the frequency parameter ω; see Fig. 11.5.2. If the sensor and actuator are noncolocated, the above-mentioned pole-zero alternation in general does not exist.

Solution by the Toolbox The Toolbox of this offers a chapter function pzTF for computing the poles and zeros of transfer function $G_p(s)$ and for displaying the modal expansion of the transfer function; see Window 5.3.

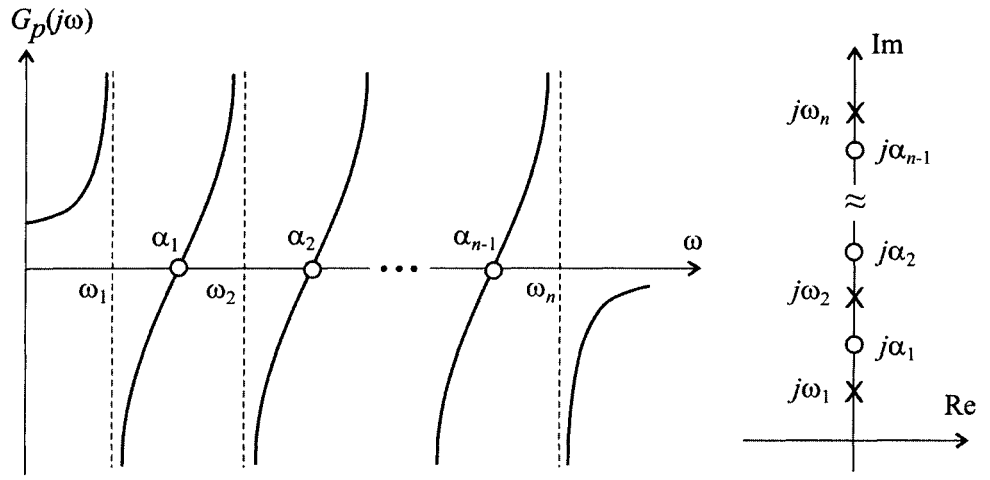


FIGURE 11.5.2 Alternation of poles and zeros.

Window 5.3. Function pzTF

MATLAB Function: **pzTF**

Purpose:
To display transfer function $G_p(s)$ and to compute its poles and zeros.

Synopsis:

```
pzTF(Cs, Bc)
[num, den, p, z, modcoeff] = pzTF(Cs, Bc)
```

Description:
`pzTF(Cs, Bc)` displays transfer function $G_p(s)$ in a modal series and its poles and zeros, where `Cs` and `Bc` specify the sensor and actuator locations with the options:

`Cs` = integer j	assign the sensor at the station corresponding to x_j
`Cs` = n-vector	assign the sensor distribution vector $\{C_s\}$ from Eq. (11.100b)
`Bc` = integer k	assign the actuator at the station corresponding to x_k
`Bc` = n-vector	assign the actuator distribution vector $\{B_c\}$ in Eq. (11.100a)

Furthermore, if the mechanical system is undamped or proportionally damped, the modal series (11.103) is also displayed.

`[num, den, p, z, modcoeff] = pzTF(Cs, Bc)` returns the coefficients of the numerator and denominator of $G_p(s)$ in vectors `num` and `den`, respectively; the poles and zeros of $G_p(s)$ in vectors `p` and `z`, respectively; and the coefficients in the modal series (11.103) for an undamped or proportionally damped system in a four-row matrix `modcoeff`. The first row of `modcoeff` stores the coefficients a_k, the second row b_k, the third row c_k, and the fourth row ω_k.

EXAMPLE 5.2

Consider a 3-DOF undamped mechanical system with parameter matrices

$$[M] = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 60 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \quad [K] = \begin{bmatrix} 20 & -10 & -10 \\ -10 & 50 & -20 \\ -10 & -20 & 30 \end{bmatrix}$$

The system is controlled via a sensor and an actuator, both located at the first station corresponding to $x_1(t)$. This is a colocated control system. By

```
>> m=diag([50 60 100]); k=10*[2 -1 -1; -1 5 -1; -1 -1 3];
>> setmdof(m, k)
>> pzTF(1,1)
```

the transfer function $X_1(s)/U(s)$ is obtained in a modal series as follows

```
Modal expansion of transfer function
   G(s) = Y(s)/U(s) = Term 1 + Term 2 + ···
```

```
where
                d1
   Term 1 = ---------- , with d1 = 0.007194, w1 = 0.36735
            s^2 + w1^2
                d2
   Term 2 = ---------- , with d2 = 0.010978, w2 = 0.69918
            s^2 + w2^2
                d3
   Term 3 = ---------- , with d3 = 0.0018279, w3 = 0.9537
            s^2 + w3^2
```

Also, the poles and zeros of the transfer function are plotted in Fig. 11.5.3, which shows pole-zero alternation.

If the sensor is moved to the third station, the control system becomes noncolocated. In this case,

```
>> pzTF(3,1)
```

plots the poles and zeros of $X_3(s)/U(s)$ in Fig. 11.5.4. As can be seen, poles and zeros are not alternated.

Now introduce Rayleigh damping $[C] = 0.1 \cdot [M] + 0.01 \cdot [K]$ into the mechanical system. Instead of assigning the positions for the sensor and actuator, assume the distribution vectors $\{C_s\}$ and $\{B_c\}$ in the following two cases:

(a) Colocated system: $\{C_s\} = \{B_c\} = \begin{pmatrix} 1 & 2 & 0 \end{pmatrix}^T$; and

(b) Noncolocated control system: $\{C_s\} = \begin{pmatrix} 1 & 2 & 0 \end{pmatrix}^T$, $\{B_c\} = \begin{pmatrix} -1 & 1 & 1 \end{pmatrix}^T$.

By the commands

```
>> setmdof(m, k, 0.1*m + 0.01*k)   % introducing damping
>> pzTF([1; 2; 0], [1; 2; 0])      % Case 1
>> pzTF([1; 2; 0], [-1; 1; 1])     % Case 2
```

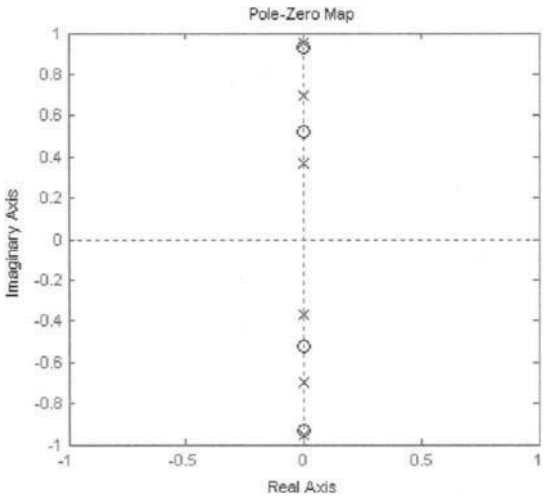


FIGURE 11.5.3 Transfer function poles and zeroes: colocated system without damping.

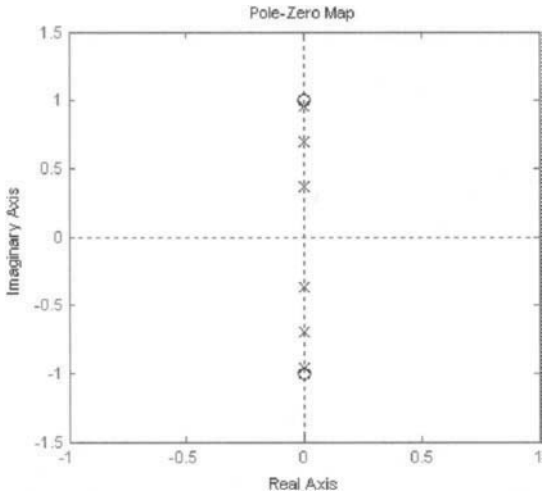


FIGURE 11.5.4 Transfer function poles and zeroes: noncolocated system without damping.

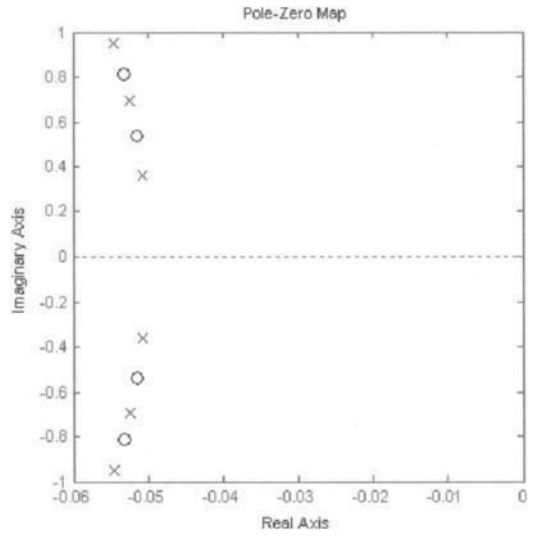


FIGURE 11.5.5 Case (a) colocated system with Rayleigh damping.

the poles and zeros of the open-loop transfer function are plotted in Figs. 11.5.5 and 11.5.6. While the transfer function poles in both the cases are the same, noncolocation of the sensor and actuator dramatically changes the locations of the transfer function zeros. Furthermore, by the commands

```
>> % Case (a) Colocated system
>> [num1, den1] = pzTF([1; 2; 0], [1; 2; 0]);
>> sys1 = tf(num1, den1);
>> bode(sys1); grid
```

```
>> % Case (b) Noncolocated system
>> [num2, den2] = pzTF([1; 2; 0], [-1; 1; 1]) ;
>> sys2 = tf(num2, den2);
>> bode(sys2); grid
```

the bode diagram of the colocated transfer function is plotted in Fig. 11.5.7, which shows a minimum phase, and the bode diagram of the noncolocated transfer function is plotted in Fig. 11.5.8, which does not have nonminimum phase.

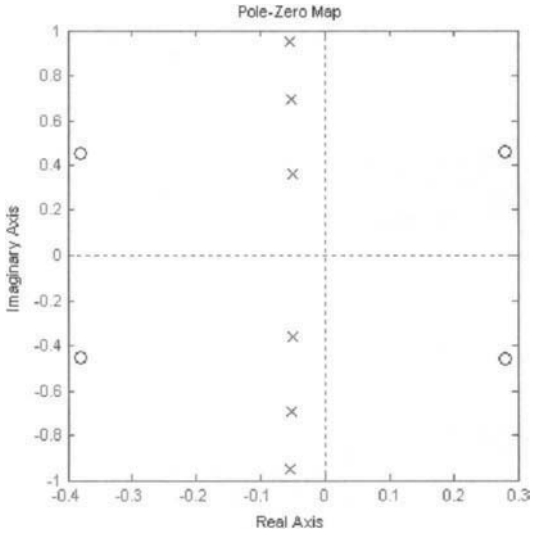


FIGURE 11.5.6 Case (b) noncolocated system with Rayleigh damping.

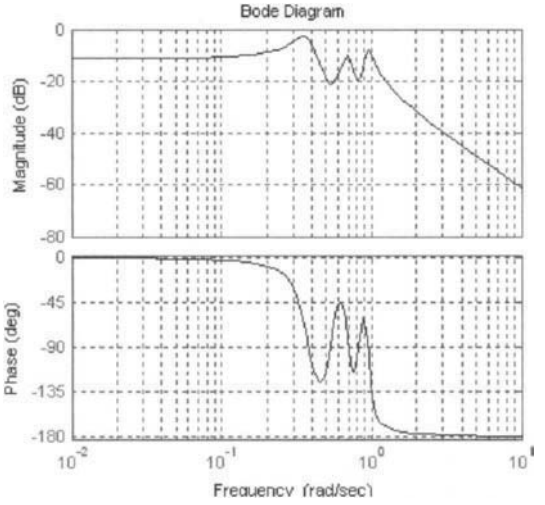


FIGURE 11.5.7 Case (a) colocated system.

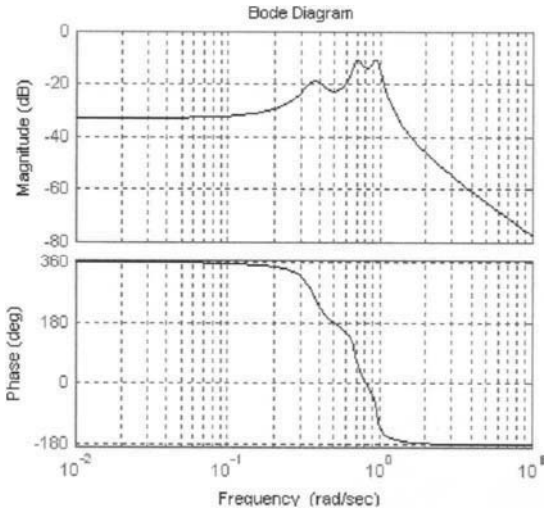


FIGURE 11.5.8 Case (b) noncolocated system.

11.6 Feedback Control

With technology advancements in sensors, actuators and digital computers, vibration control is also achieved through the use of actively adjusted external forces that are applied according to some feedback control logic and measurements of vibration response. This kind of technique is known as *active vibration control methods*.

The current section is about active control of mechanical systems. The mechanical system (plant) in consideration is a linear system of multiple degrees of freedom. Depending on applications, a feedback controller is added to the plant for vibration control (Section 11.6.1), or for position control (Section 11.6.2). As shall be seen, the objectives and formulation for a vibration control system are different from those for a position control system.

11.6.1 Vibration Control System

A typical vibration control system is shown in Fig. 11.6.1. The purpose of adding a feedback controller to the mechanical system is to suppress the vibration response of the system under external and/or initial disturbances.

Governing Equations Assume that the feedback controller has p sensors and q actuators. The mechanical system under such a controller is governed by the differential equation

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \{B_e\}f_e(t) + [B_c]\{u(t)\} \tag{11.108}$$

where $f_e(t)$ is an external load, $\{B_e\}$ is a load distribution vector, $\{u(t)\}$ is the vector of the q control forces applied by the actuators

$$\{u(t)\} = \begin{pmatrix} u_1(t) & u_2(t) & \cdots & u_q(t) \end{pmatrix}^T \tag{11.109}$$

and $[B_c]$ is an n-by-q matrix. Write the vector of p sensor signals as

$$\{y(t)\} = \begin{pmatrix} y_1(t) & y_2(t) & \cdots & y_p(t) \end{pmatrix}^T \tag{11.110}$$

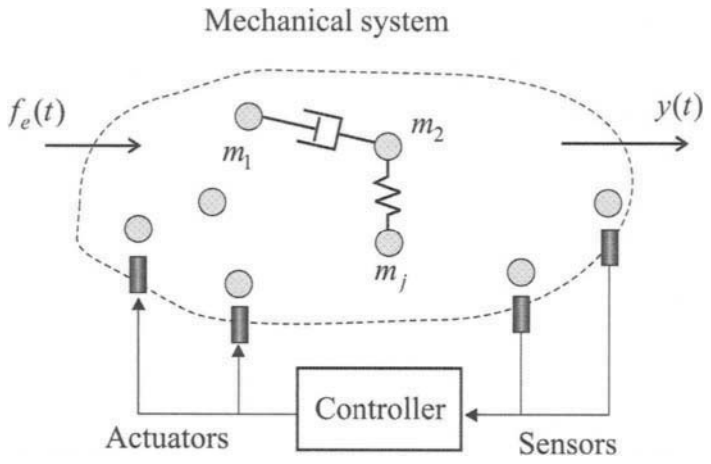


FIGURE 11.6.1 Schematic of a vibration control system.

Let the sensors measure either displacements and/or velocities. The sensor output signals can be expressed by

$$\{y(t)\} = [E_0]\{x(t)\} + [E_1]\{\dot{x}(t)\} \tag{11.111}$$

where $[E_0]$ and $[E_1]$ are the constant matrices.

State Representation Equation (11.108) is cast into

$$\{\dot{z}(t)\} = [A]\{z(t)\} + [B]\{u(t)\} + \{F(t)\} \tag{11.112}$$

where the state vector

$$\{z(t)\} = \begin{Bmatrix} \{x(t)\} \\ \{\dot{x}(t)\} \end{Bmatrix} \tag{11.113}$$

and

$$[A] = \begin{bmatrix} [0] & [I] \\ -[M]^{-1}[K] & -[M]^{-1}[C] \end{bmatrix}, \quad [B] = \begin{bmatrix} [0] \\ [M]^{-1}[B_c] \end{bmatrix},$$
$$\{F(t)\} = \begin{Bmatrix} \{0\} \\ [M]^{-1}\{B_e\} f_e(t) \end{Bmatrix} \tag{11.114}$$

with [0] and {0} being zero matrix and vector of appropriate order, respectively. The sensor output equation (11.111) is also cast in a state form

$$\{y(t)\} = [E]\{z(t)\} \tag{11.115}$$

where

$$[E] = \begin{bmatrix} [E_0] & [E_1] \end{bmatrix} \tag{11.116}$$

Feedback Controller The objective of controller design is to suppress the vibration response $\{x(t)\}$ of the mechanical system. The following three feedback control laws are considered:

(i) State feedback

$$\{u(t)\} = -[K_c]\ \{z(t)\} \tag{11.117}$$

where $[K_c]$ is a control gain matrix to be designed. The closed-loop system is governed by

$$\{\dot{z}(t)\} = \big([A] - [B]\,[K_c]\big)\,\{z(t)\} + \{F(t)\} \tag{11.118}$$

which is obtained by substituting Eq. (11.117) into Eq. (11.112). The control gain matrix is normally obtained by pole assignment or linear quadratic regulator (LQR) method.

(ii) Direct output feedback

$$\{u(t)\} = -[K_c]\,\{y(t)\} = -[K_c]\,[E]\,\{z(t)\} \tag{11.119}$$

where $\{y(t)\}$ is the vector of output signals, and $[K_c]$ is a control gain matrix to be designed. In this case, the closed-loop system is governed by

$$\{\dot{z}(t)\} = \big([A] - [B]\,[K_c]\,[E]\big)\,\{z(t)\} + \{F(t)\} \tag{11.120}$$

(iii) PID feedback with one sensor and one actuator
Assume that the sensor measures the jth displacement $x_j(t)$, and that the actuator applies a force to the mechanical system at the station corresponding to the kth displacement $x_k(t)$. Under PID (proportional-integral-derivative) feedback, the control force vector becomes a scalar

$$\{u(t)\} = u(t) = -\left(K_p y(t) + K_i \int_{t_0}^{t} y(\tau)\,d\tau + K_d \frac{d}{dt} y(t)\right) \tag{11.121}$$

where K_p, K_i, and K_d are the gain parameters of the controller to be designed, and $y(t)$ is the sensor output given by

$$y(t) = x_j(t) = \{e_j\}^T\,\{x(t)\} \tag{11.122}$$

with $\{e_j\}$ being an n-vector whose jth element is one and whose all other elements are zeros. The matrix $[B_c]$ in Eq. (11.108) becomes

$$[B_c] = \{e_k\} \tag{11.123}$$

The above PID-controlled mechanical system can be represented by either transfer function formulation or state formulation.

Transfer Function Formulation Substituting Eqs. (11.121) to (11.123) into Eq. (11.108) and taking Laplace transform of the resulting equation with respect to time (with zero initial conditions), yields

$$\{X(s)\} = H(s)\left[\{B_e\} F_e(s) - \{e_k\} G_c(s) Y(s)\right] \tag{11.124}$$

$$Y(s) = \frac{1}{1 + G_p(s) G_c(s)} \{e_j\}^T H(s) \{B_e\} F_e(s) \tag{11.125}$$

where $\{X(s)\}$, $Y(s)$, and $F_e(s)$ are the Laplace transforms of $\{x(t)\}$, $y(t)$, and $f_e(t)$, respectively; $H(s)$ is the transfer function of the mechanical system as given in Eq. (11.92);

$$G_p(s) = \{e_j\}^T H(s) \{e_k\}; \tag{11.126}$$

and $G_c(s)$ is the controller transfer function

$$G_c(s) = \frac{U(s)}{Y(s)} = K_p + K_i \frac{1}{s} + K_d s \tag{11.127}$$

with $U(s)$ being the Laplace transform of the control force $u(t)$. From Eq. (11.125), the characteristic equation of the control system (closed-loop) is

$$1 + G_p(s) G_c(s) = 0. \tag{11.128}$$

State-Space Formulation Introduce an auxiliary variable

$$\xi_a(t) = \int_{t_0}^{t} y(\tau)\, d\tau \tag{11.129a}$$

which indicates that

$$\dot{\xi}_a(t) = y(t) = \{e_j\}^T \{x(t)\} \tag{11.129b}$$

The control force given in Eq. (11.121) can be written as

$$\{u(t)\} = -[K_c]\{z(t)\} - K_i \xi_a(t) \tag{11.130}$$

with

$$[K_c] = \left[K_p \{e_j\}^T \quad K_d \{e_j\}^T\right] \tag{11.131}$$

Substituting Eq. (11.130) into Eq. (11.112) gives

$$\{\dot{z}(t)\} = \left([A] - [B][K_c]\right)\{z(t)\} - [B] K_i \xi_a(t) + \{F(t)\} \tag{11.132}$$

Combining Eqs. (11.132) and (11.129b) yields an augmented state equation for the control system:

$$\begin{pmatrix} \{\dot{z}(t)\} \\ \dot{\xi}_a(t) \end{pmatrix} = \begin{bmatrix} [A] - [B][K_c] & -K_i [B] \\ \left[\{e_j\}^T \quad \{0\}^T\right] & 0 \end{bmatrix} \begin{pmatrix} \{z(t)\} \\ \xi_a(t) \end{pmatrix} + \begin{pmatrix} \{F(t)\} \\ \{0\} \end{pmatrix} \tag{11.133}$$

Note. The auxiliary variable $\xi_a(t)$ is only needed when integration is involved in the control action. If $K_i = 0$, instead of Eq. (11.133), Eq. (11.118) with the gain matrix specified by Eq. (11.131) can be directly used.

Response of Controlled Mechanical System The closed-loop state Eq. (11.118), (11.120), and (11.133) can be written in the unified form

$$\{\dot{\eta}(t)\} = [A_c]\{\eta(t)\} + \{F_a(t)\} \tag{11.134}$$

where $\{\eta(t)\} = \{z(t)\}$ for the state feedback and direct output feedback controls, and $\{\eta(t)\} = \left(\{z(t)\}^T \quad \xi_a(t)\right)^T$ for the PID feedback control. With the control gains selected, the response of the control system can be computed numerically. To this end, cast Eq. (11.134) into

$$\{\dot{\eta}(t)\} = \Gamma(t,\ \{\eta(t)\}), \quad \{\eta(0)\} = \{\eta_0\} \tag{11.135}$$

with

$$\Gamma(t,\ \eta(t)) = [A_c]\{\eta(t)\} + \{F_a(t)\} \tag{11.136}$$

Equation (11.135) is solved by a fixed-step Runge-Kutta algorithm of order four:

$$\{\eta_{k+1}\} = \{\eta_k\} + \frac{h}{6}\left(\{g_1\} + 2\{g_2\} + 2\{g_3\} + \{g_4\}\right) \tag{11.137}$$

where

$$\begin{aligned}
\{g_1\} &= \Gamma\left(t_k, \{\eta_k\}\right) \\
\{g_2\} &= \Gamma\left(t_k + \frac{h}{2}, \{\eta_k\} + \frac{h}{2}\{g_1\}\right) \\
\{g_3\} &= \Gamma\left(t_k + \frac{h}{2}, \{\eta_k\} + \frac{h}{2}\{g_2\}\right) \\
\{g_4\} &= \Gamma\left(t_k + h, \{\eta_k\} + h\{g_3\}\right)
\end{aligned} \tag{11.138}$$

$\{\eta_k\} = \{\eta(t_k)\}$, $t_k = kh$, $k = 0, 1, 2, \ldots$, and h is the step size in the numerical integration.

11.6.2 Position Control System

A position control system is shown in Fig. 11.6.2, where the function of the feedback controller is to assure that the output $y(t)$ of the mechanical system tracks the reference input $r(t)$ with zero or minimum error. In many cases, like servo control of flexible positioning systems, step input is considered.

Governing Equations Without loss of generality, feedback control with one sensor and one actuator is considered. Assume that the sensor measures the jth displacement $x_j(t)$, and that the actuator applies a force to the mechanical system in the direction of the kth displacement $x_k(t)$. The mechanical system with such a controller is governed by the differential equation

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \{e_k\}u(t) \tag{11.139}$$

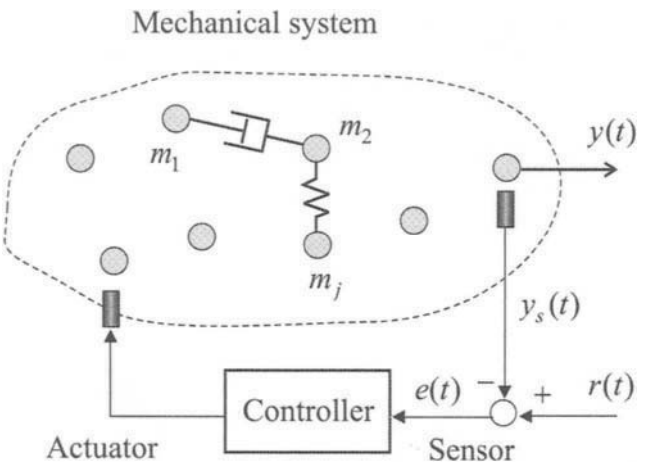


FIGURE 11.6.2 Schematic of a position control system.

where $u(t)$ is the control force applied to the mechanical system by the actuator, and $\{e_k\}$ is an n-vector with kth element being one and the rest zeros. The output of the mechanical system is

$$y(t) = x_j(t) = \{e_j\}^T \{x(t)\} \tag{11.140}$$

The sensor output (measured system output) is given by

$$y_s(t) = k_s y(t) = k_s \{e_j\}^T \{x(t)\} \tag{11.141}$$

where k_s is a constant sensor gain. The controller input is the error signal

$$e(t) = r(t) - y_s(t) \tag{11.142}$$

Transfer Function Formulation Laplace transform of Eqs. (11.139), (11.140), and (11.142) with respect to time yields

$$\{X(s)\} = H(s)\{e_k\} U(s) \tag{11.143}$$

$$Y(s) = \{e_j\}^T \{X(s)\} \tag{11.144}$$

$$E(t) = R(s) - k_s Y(s) \tag{11.145}$$

where $H(s) = \left(s^2 [M] + s [C] + [K]\right)^{-1}$. Write the controller transfer function as

$$G_c(s) = \frac{U(s)}{E(s)} = T_a(s) T_c(s) \tag{11.146}$$

where $T_c(s)$ represents a control logic and $T_a(s)$ is the actuator transfer function. It follows from Eqs. (11.143) to (11.146) that the closed-loop transfer function is

$$\frac{Y(s)}{R(s)} = \frac{G_c(s) G_p(s)}{1 + k_s G_c(s) G_p(s)} \tag{11.147}$$

where the transfer function of the plant is given by

$$G_p(s) = \{e_j\}^T H(s) \{e_k\} \tag{11.148}$$

which has been studied in Section 11.5.2. From Eq. (11.147), the closed-loop characteristic equation is

$$1 + k_s G_c(s) G_p(s) = 0. \tag{11.149}$$

State Representation Equation (11.139) is cast into

$$\{\dot{z}(t)\} = [A] \{z(t)\} + [B] u(t) \tag{11.150}$$

where $[A]$ and $\{z(t)\}$ are given in Eqs. (11.113) and (11.114), and [B] is of the form

$$[B] = \begin{bmatrix} \{0\} \\ [M]^{-1} \{e_k\} \end{bmatrix} \tag{11.151}$$

with $\{0\}$ being a zero vector. The output of the control system is

$$y(t) = [\Psi_s] \{z(t)\} \tag{11.152}$$

where

$$[\Psi_s] = \begin{bmatrix} \{e_j\}^T & \{0\}^T \end{bmatrix} \tag{11.153}$$

The error signal is expressed by

$$e(t) = r(t) - k_s [\Psi_s] \{z(t)\} \tag{11.154}$$

Feedback Controller Three feedback control laws are considered.

(i) PD feedback

$$G_c(s) = \frac{U(s)}{E(s)} = K_p + K_d s \tag{11.155}$$

which indicates that

$$u(t) = K_p e(t) + K_d \dot{e}(t) = -[K_c] \{z(t)\} + K_p r(t) + K_d \dot{r}(t) \tag{11.156}$$

where

$$[K_c] = k_s \begin{bmatrix} K_p \{e_j\}^T & K_d \{e_j\}^T \end{bmatrix} \tag{11.157}$$

Substitute Eq. (11.156) into Eq. (11.150) to obtain the state equation of the closed-loop

$$\{\dot{z}(t)\} = ([A] - [B][K_c]) \{z(t)\} + K_p [B] r(t) + K_d [B] \dot{r}(t) \tag{11.158}$$

(ii) PID feedback

$$G_c(s) = \frac{U(s)}{E(s)} = K_p + K_i \frac{1}{s} + K_d s \tag{11.159}$$

or

$$u(t) = K_p e(t) + K_i \int_{t_0}^{t} e(\tau)\, d\tau + K_d \frac{d}{dt} e(t) \tag{11.160}$$

Introduce an auxiliary variable

$$\xi_a(t) = \int_{t_0}^{t} e(\tau)\, d\tau \tag{11.161}$$

which implies that

$$\dot{\xi}_a(t) = r(t) - k_s \left[\Psi_s\right] \{z(t)\} \tag{11.162}$$

Following the PID feedback control in Section 11.6.1, it can be shown that

$$u(t) = -\left[K_c\right] \{z(t)\} + K_i \xi_a(t) + K_p r(t) + K_d \dot{r}(t) \tag{11.163}$$

with $[K_c]$ given in Eq. (11.157). Substituting Eq. (11.163) into Eq. (11.150) and combining the resulting equation with Eq. (11.162) gives an augmented state equation for the control system:

$$\begin{pmatrix} \{\dot{z}(t)\} \\ \dot{\xi}_a(t) \end{pmatrix} = \begin{bmatrix} [A] - [B]\left[K_c\right] & K_i [B] \\ -k_s \left[\Psi_s\right] & 0 \end{bmatrix} \begin{pmatrix} \{z(t)\} \\ \xi_a(t) \end{pmatrix} + \begin{pmatrix} [B]\left(K_p r(t) + K_d \dot{r}(t)\right) \\ r(t) \end{pmatrix} \tag{11.164}$$

(iii) General feedback control law

$$G_c(s) = \frac{U(s)}{E(s)} = \frac{b_{m+1} s^{m+1} + b_m s^m + \cdots + b_1 s + b_0}{a_m s^m + a_{m-1} s^{m-1} + \cdots + a_1 s + a_0} \tag{11.165}$$

where $a_m \neq 0$. By long division, the above transfer function can be written as

$$G_c(s) = K_p + K_d s + \tilde{G}_c(s) \tag{11.166}$$

where

$$\tilde{G}_c(s) = \frac{\tilde{b}_{m-1} s^{m-1} + \cdots + \tilde{b}_1 s + \tilde{b}_0}{s^m + \tilde{a}_{m-1} s^{m-1} + \cdots + \tilde{a}_1 s + \tilde{a}_0} \tag{11.167}$$

with

$$\tilde{a}_k = \frac{a_k}{a_m}, \quad k = 0, 1, \ldots, m-1. \tag{11.168}$$

Write

$$U(s) = U_1(s) + U_2(s) \tag{11.169}$$

where

$$\begin{aligned} U_1(s) &= (K_p + K_d s)E(s) \\ U_2(s) &= \tilde{G}_c(s)E(s) \end{aligned} \tag{11.170}$$

According to Eq. (11.156),

$$u_1(t) = -[K_c]\{z(t)\} + K_p r(t) + K_d \dot{r}(t) \tag{11.171}$$

A state representation of the input-output relation $U_2(s) = \tilde{G}_c(s)E(s)$ is

$$\begin{aligned} \{\dot{x}_a(t)\} &= [A_a]\{x_a(t)\} + [B_a]e(t) \\ u_2(t) &= [C_a]\{x_a(t)\} \end{aligned} \tag{11.172}$$

where $\{x_a(t)\}$ is a state vector of m auxiliary variables, and the error signal is given by Eq. (11.154). The matrices in Eq. (11.172) may be constructed in different ways. For instance, one can use the controllable canonical form (Reference 8)

$$[A_a] = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -\tilde{a}_0 & -\tilde{a}_1 & \cdots & -\tilde{a}_{m-2} & -\tilde{a}_{m-1} \end{bmatrix}, \quad [B_a] = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{11.173}$$

$$[C_a] = \begin{bmatrix} \tilde{b}_0 & \tilde{b}_2 & \cdots & \tilde{b}_{m-2} & \tilde{b}_{m-1} \end{bmatrix}$$

It follows that

$$u(t) = u_1(t) + u_2(t) = -[K_c]\{z(t)\} + [C_a]\{x_a(t)\} + K_p r(t) + K_d \dot{r}(t) \tag{11.174}$$

Substitute Eq. (11.174) into Eq. (11.150) and combine the result with the first of Eqs. (11.172) to obtain the augmented state equation for the control system:

$$\begin{Bmatrix} \{\dot{z}(t)\} \\ \{\dot{x}_a(t)\} \end{Bmatrix} = \begin{bmatrix} [A] - [B][K_c] & [B][C_a] \\ -k_s[B_a][\Psi_s] & [A_a] \end{bmatrix} \begin{Bmatrix} \{z(t)\} \\ \{x_a(t)\} \end{Bmatrix} + \begin{Bmatrix} [B]\left(K_p r(t) + K_d \dot{r}(t)\right) \\ [B_a] r(t) \end{Bmatrix} \tag{11.175}$$

Response of Controlled Mechanical System The closed-loop state equations (11.158), (11.164), and (11.175) can be written in the unified form (11.134). Once the control gains are

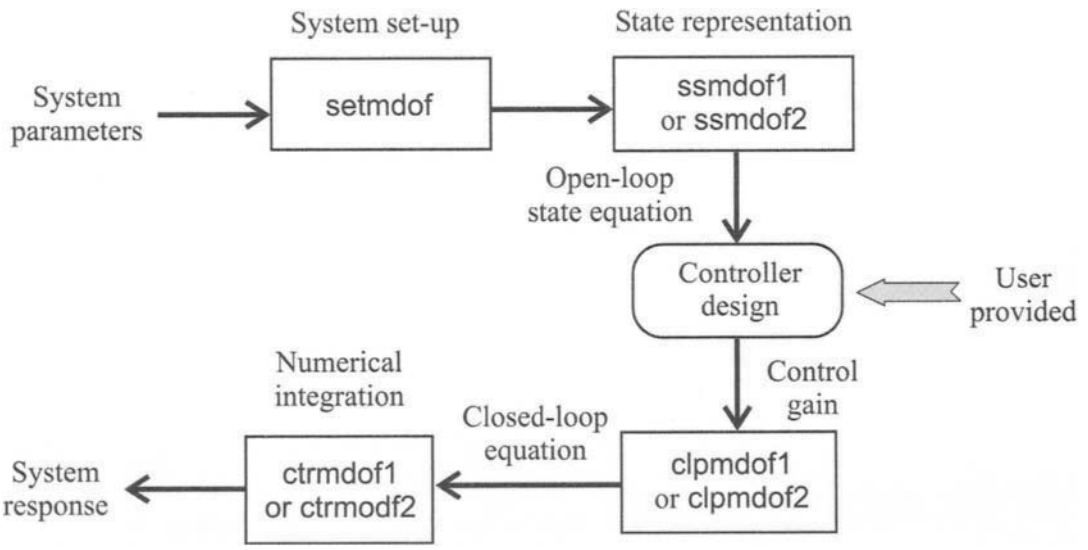


FIGURE 11.6.3 A flowchart of control system formulation and simulation by the Toolbox.

selected, these state equations can be solved by numerical integration like the Runge-Kutta algorithm given in Eq. (11.137).

11.6.3 Solution by the Toolbox

The Toolbox of this chapter offers several MATLAB functions for formation and simulation of vibration and position control systems. The solution procedure by the Toolbox takes the following five steps:

Step 1. Specify the mechanical system by function `setmdof`;
Step 2. Obtain matrices [*A*] and [*B*] of the state equation (11.112) or (11.150) by function `ssmdof1` or `ssmdof2`;
Step 3. Determine the controller gains for one of the control laws described in the previous sections;
Step 4. With the selected control gains, obtain the matrices of the closed-loop state equation by function `clpmdof1` or `clpmdof2`; and
Step 5. Simulate the response of the control system by function `ctrmdof1` or `ctrmdof2`.

Here, functions `ssmdof1`, `clpmdof1`, and `ctrmdof1` are for a vibration control system as described in Section 11.6.1, while functions `ssmdof2`, `clpmdof2`, and `ctrmdof2` are for a position control system as described in Section 11.6.2.

The above solution procedure is illustrated in a flowchart in Fig. 11.6.3. Step 1 is the same as the setup for dynamic analysis described in Section 11.2.3. For Step 3 (controller design), the reader may refer to Appendix B7 for some useful M-file functions from the MATLAB Control System Toolbox. Moreover, functions `getTF` and `pzTF` introduced in Section 11.5 and function `frfun` introduced in Section 11.3.6 may be useful in controller design.

State Representation This Toolbox has functions `ssmdof1` and `ssmdof2` for obtaining the state equation for an M-DOF mechanical system; see Window 6.1.

Window 6.1. Functions ssmdof1 and ssmdof2

MATLAB Functions: **ssmdof1, ssmdof2**

Purpose:
To obtain a state representation for an M-DOF mechanical system.

Synopsis:

```
ssmdof1(B_actuators)
[A, B] = ssmdof1(B_actuators)
ssmdof2(j_actuator, j_sensor, sensor_gain)
[A, B, Phi_s] = ssmdof2(j_actuator, j_sensor, sensor_gain)
```

Description:
`ssmdof1(B_actuators)` computes and displays matrices $[A]$ and $[B]$ in the state equation (11.112) for a vibration control system. Here `B_actuators` is a vector of integers indicating the locations of the actuators. For instance, `B_actuators = [3 1 5]` indicates that there are three actuators, which are located in the direction of the displacements x_3, x_1, and x_5, respectively. By default, `B_actuators = 1`.

`[A, B] = ssmdof1(B_actuators)` returns the above-mentioned matrices $[A]$ and $[B]$ in `A` and `B`, respectively.

`ssmdof2(j_actuator, j_sensor, sensor_gain)` computes and displays matrices $[A]$, $[B]$ in the state equation (11.150) and matrix $[\Psi_s]$ in the output equation (11.152), for a position control system. Here `j_actuator` and `j_sensor` are integers indicating the actuator and sensor locations, and `sensor_gain` is the sensor gain k_s defined in Eq. (11.141). For instance, `j_actuator = 2` indicates that the actuator is located in the direction of the displacement x_2, and `j_sensor = 3` indicates that the sensor measures the displacement x_3. By default, `j_actuator = 1`, `j_sensor = 1`, and `sensor_gain = 1`.

`[A, B, Phi_s] = ssmdof2(j_actuator, j_sensor, sensor_gain)` returns the above-mentioned matrices $[A]$, $[B]$, and $[\Psi_s]$ in `A`, `B`, `Phi_s`, respectively.

EXAMPLE 6.1

In Fig. 11.6.4, a spring-mass system is under a feedback controller with a sensor measuring the displacement of the second mass and an actuator applying a control force to the first mass. This is a SISO position control system as presented in Section 11.6.2, with $r(t)$ being the command input and x_2 being the output. The equations of motion of the mechanical system are

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{bmatrix} c & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} u$$

where u is the control force applied to the first mass by the actuator. The sensor equation, by Eq. (11.141), is

$$y_s(t) = k_s x_2(t)$$

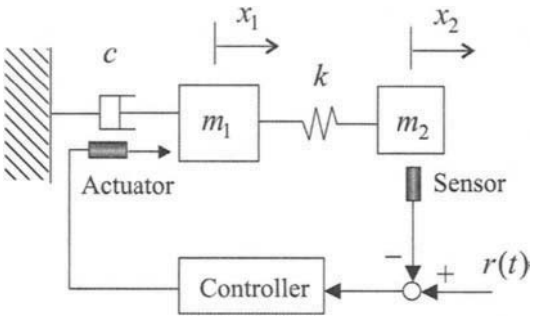


FIGURE 11.6.4 Position control of a spring-mass system.

Let the parameters of the mechanical system be $m_1 = 1.5$, $m_2 = 0.2$, $k = 50$, and $c = 0.25$. Assume that the sensor gain $k_s = 1$. The commands

```
>> M = [1.5 0; 0 0.2]; C = [0.25 0; 0 0]; K = [50 -50; -50 50];
>> setmdof(M,K,C)
>> ssmdof2(2,1)
```

yield the following results

```
State equation: d{z}/dt = [A]*{z} + [B]*u, order of {z} = 4

Sensor output equation: ys = ks*[Phi_s]*{z}, with gain ks = 1

Matrix [A]:
      0              0          1.0000e+000       0
      0              0                0       1.0000e+000
-3.3333e+001    3.3333e+001   -1.6667e-001       0
 2.5000e+002   -2.5000e+002          0           0

Matrix [B]:
  0
  0
  0
  5

Matrix [Phi_s]:
  1     0     0     0
```

Also,

```
>> getTF([1;0])
```

gives the transfer function $G_p(s) = X_2(s)/U(s)$ as follows

```
                  166.7
-------------------------------------
s^4 + 0.1667 s^3 + 283.3 s^2 + 41.67 s
```

Closed-Loop Formulation Once a state representation for the mechanical system is obtained, the MATLAB Control System Toolbox can be used for study of the properties of the control system (e.g., stability, controllability, and observability), and for design of a feedback controller. With the selected control gains, the state equation of the closed-loop system can be obtained. This Toolbox offers functions `clpmdof1` and `clpmdof2` for obtaining the closed-loop state equation; see Windows 6.2 and 6.3.

Window 6.2. Function `clpmdof1`

MATLAB Function: **`clpmdof1`**

Purpose:
To obtain the closed-loop state equation for a vibration control system.

Synopsis:

```
clpmdof1(ID_control, K_gain, S_Info)
Ac = clpmdof1(ID_control, K_gain, S_Info)
```

Description:
`clpmdof1(ID_control, K_gain, S_Info)` obtains the closed-loop state equation of a mechanical system under a feedback controller for vibration control. Here, `ID_control` is an integer identifying one of the three feedback controllers given in Section 11.6.1, and `K_gain` and `S_Info` are gain matrices as explained below.

`ID_control` = 1: state feedback $\{u(t)\} = -[K_c]\{z(t)\}$. The closed-loop system is described by Eq. (11.118). `K_gain` is the control gain matrix $[K_c]$. `S_Info` is not assigned. In other words, `clpmdof1(ID_control, K_gain)` is adequate.

`ID_control` = 2: output feedback $\{u(t)\} = -[K_c]\{y(t)\}$. The closed-loop system is described by Eq. (11.120). `K_gain` is the control gain matrix $[K_c]$, and `S_Info` is matrix $[E]$ in Eq. (11.115).

`ID_control` = 3: PID feedback with one sensor and one actuator

$$\{u(t)\} = -\left(K_p y(t) + K_i \int_{t_0}^{t} y(\tau)\, d\tau + K_d \frac{d}{dt} y(t)\right).$$

The closed-loop system is described by the augmented equation (11.133). In this case, `K_gain = [Kp Ki Kd]`, where `Kp`, `Ki`, and `Kd` are the controller parameters, and `S_Info` is an integer representing the sensor location.

`Ac = clpmdof1(ID_control, K_gain, S_Info)` returns the state matrix of the closed-loop system in `Ac`.

Note:
Before `clpmdof1` can be used, function `ssmdof1` must be executed.

Window 6.3. Function clpmdof2

MATLAB Function: **clpmdof2**

Purpose:
To obtain the closed-loop state equation for a position control system.

Synopsis:

```
clpmdof2(ID_control, K_gain)
Ac = clpmdof2(ID_control, K_gain)
```

Description:
`clpmdof2(ID_control, K_gain)` obtains the closed-loop state equation of a mechanical system under a feedback controller for position control. Here, `ID_control` is an integer identifying one of the three feedback controllers given in Section 11.6.2, and `K_gain` is a gain matrix as explained below.

`ID_control` = 1: PD feedback $u(t) = K_p e(t) + K_d \dot{e}(t)$. The closed-loop system is described by Eq. (11.158). `K_gain = [Kp Kd]`, where `Kp` and `Kd` are the controller parameters.

`ID_control` = 2: PID feedback $u(t) = K_p e(t) + K_i \int_{t_0}^{t} e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$. The closed-loop system is described by Eq. (11.164). `K_gain = [Kp Ki Kd]`, where `Kp`, `Ki`, and `Kd` are the controller parameters.

`ID_control` = 3: general feedback control law given in Eq. (11.165), i.e.,

$$G_c(s) = \frac{U(s)}{E(s)} = \frac{b_{m+1}s^{m+1} + b_m s^m + \cdots + b_1 s + b_0}{a_m s^m + a_{m-1}s^{m-1} + \cdots + a_1 s + a_0}, \quad a_m \neq 0.$$

The closed-loop system is described by the augmented equation (11.175). In this case, `K_gain` is a matrix in one of the following two forms

(i) for $b_{m+1} \neq 0$, `K_gain` $= \begin{bmatrix} b_{m+1} & b_m & \cdots & b_1 & b_0 \\ 0 & a_m & \cdots & a_1 & a_0 \end{bmatrix}$;

(ii) for $b_{m+1} = 0$, `K_gain` $= \begin{bmatrix} b_m & \cdots & b_1 & b_0 \\ a_m & \cdots & a_1 & a_0 \end{bmatrix}$.

`Ac = clpmdof2(ID_control, K_gain)` returns the matrix of the closed-loop state equation in `Ac`.

Note:
Before `clpmdof2` can be used, function `ssmdof2` must be executed.

EXAMPLE 6.2

Consider the same control system as given in Example 6.1. Consider PI control

$$G_c(s) = 2 + \frac{0.03}{s}$$

which has the form of Eq. (11.159). Assume that a state representation of the mechanical system has been obtained by function ssmdof2, as shown in Example 6.1. The command

```
>> clpmdof2(2, [2 0.03 0])
```

yields the following results

```
Control System Formulation
--------------------------
                                                t
 Control law: PID feedback u = - Kp*e(t) - Ki*int{e(t)}*dt -Kd*de(t)/dt
                                               t0
 Gain parameters:  Kp = 2, Ki = 0.03, Kd = 0
 error signal e(t) =  r(t)-y(t), with  y(t) = x_1
 sensor location at x_1, actuator location at x_2
Closed-loop state equation d{za}/dt = [Ac]*za + {Fa(t)}
 where {za} = ({z}; xi_a}, with xi_a being an auxiliary variable
 for I control action
Matrix [Ac] of closed-loop state equation:
          0           0  1.0000e+000           0           0
          0           0           0 1.0000e+000           0
-3.3333e+001  3.3333e+001 -1.6667e-001           0           0
 2.4000e+002 -2.5000e+002           0           0 1.5000e-001
-1.0000e+000           0           0           0           0
Eigenvalues of [Ac]
-9.5668e-003 +1.6797e+001i
-9.5668e-003 -1.6797e+001i
-6.6254e-002 +1.0840e+000i
-6.6254e-002 -1.0840e+000i
-1.5025e-002
```

Response of Closed-Loop With the closed-loop obtained by function clpmdof1 or clpmdof2, the response of the control system can be computed by the Runge-Kutta algorithm as given in Eq. (11.137). For this, the Toolbox offers function ctrmdof1 (Window 6.4) for a vibration control system and function ctrmdof2 (Window 6.5) for a position control system.

Window 6.4. Function ctrmdof1

MATLAB Function: **ctrmdof1**

Purpose:
To compute the response of a vibration control system by the Runge-Kutta algorithm given in Eqs. (11.137) and (11.138).

Synopsis:

```
ctrmdof1(Load_Spec, Be)
ctrmdof1(Load_Spec, Be, IC, tf, npts)
[x,t,v,u] = ctrmdof1(Load_Spec, Be, IC, tf, npts)
```

Window 6.4. Function ctrmdof1 (*Continued*)

Description:
`ctrmdof1(Load_Spec, Be)` plots the response of a vibrating control system under an external force, where `Load_Spec` is a vector specifying the external load as follows:

Impulse load $f_e(t) = I_0\delta(t)$:	`Load_Spec = [0 I0]`
Step load $f_e(t) = F_0$:	`Load_Spec = [1 F0]`
Ramp load $f_e(t) = a_0 t$:	`Load_Spec = [2 a0]`
Exponential load $f_e(t) = ae^{bt}$:	`Load_Spec = [3 a b]`
Sinusoidal load $f_e(t) = A\sin(\omega t + \phi_0)$, with ω in rad/s and ϕ_0 in degrees:	`Load_Spec = [4 A ω φ0]`
Zero load $f_e(t) = 0$	`Load_Spec = 0`

and `Be` is the load distribution vector $\{B_e\}$ in Eq. (11.108).

`ctrmdof1(Load_Spec, Be, IC, tf, npts)` plots the response of the control system, where `IC = [x0 v0]` with `x0` being the vector of initial displacements and `v0` the vector of initial velocities; `tf` is total time of simulation; and `npts` is the number of equally-spaced points in the time region `[0, tf]` for simulation. For zero initial disturbances, `IC = 0`. By default, `Load_Spec = [0 1]`, `Be` is an n-vector with the first element being one and the rest zeros, `IC = 0`, `tf` is about four times the first natural period of the mechanical system, and `npts = 1001`. Any of the parameters `Load_Spec`, `Be`, `IC`, `tf`, `npts` can be replaced by `[]` as a placeholder for its default value.

`[x,t,v,u] = ctrmdof1(Load_Spec, Be, IC, tf, npts)` returns the displacements and velocities of the controlled mechanical system in matrices `x` and `v`, times of simulation in a vector `t`, and the control force in a matrix `u`. With these output arguments, the response of the control system versus time can be plotted by the MATLAB function `plot` and by function `plotvib` of this Toolbox (see Window 3.8). For instance, both commands `plot(t, x(2,:))` and `plotvib(x,2)` plot the time history of the displacement $x_2(t)$.

Note:
Before function `ctrmdof1` can be used, function `clpmdof1` must be executed.

Window 6.5. Function ctrmdof2

MATLAB Function: **ctrmdof2**

Purpose:
To compute the response of a position control system by the Runge-Kutta algorithm given in Eqs. (11.137) and (11.138).

Synopsis:

```
ctrmdof2(Load_Spec, IC)
ctrmdof2(Load_Spec, IC, tf, npts)
[x,t,v,u,e] = ctrmdof2(Load_Spec, IC, tf, npts)
```

Window 6.5. Function ctrmdof2 (*Continued*)

Description:
`ctrmdof2(Load_Spec)` plots the response of a position control system, where `Load_Spec` is a vector specifying the reference input $r(t)$ in the following options:

Step input $r(t) = r_0$:	`Load_Spec = [1 r0]`
Ramp input $r(t) = a_0 t$:	`Load_Spec = [2 a0]`
Exponential input $r(t) = ae^{bt}$:	`Load_Spec = [3 a b]`
Zero input $r(t) = 0$:	`Load_Spec = 0`

`ctrmdof2(Load_Spec, IC, tf, npts)` plots the response of the control system, where `IC = [x0 v0]` with `x0` being the vector of initial displacements and `v0` the vector of initial velocities; `tf` is total time of simulation; and `npts` is the number of equally-spaced points in the time region `[0, tf]` for simulation. For zero initial disturbances, `IC = 0`. By default, `Load_Spec = [1 1]`, `IC = 0`, `tf` is about four times the first natural period of the mechanical system and `npts = 1001`. Any of the `Load_Spec`, `IC`, `tf`, `npts` can be replaced by `[]` as a placeholder for its default value.

`[x,t,v,u,e] = ctrmdof2(Load_Spec, IC, tf, npts)` returns the displacements and velocities of the controlled mechanical system in matrices `x` and `v`, the times of simulation in a vector `t`, the control force in a vector `u`, and error signal in a vector `e`. With these output arguments, the response of the control system versus time can be plotted by the MATLAB function `plot` and by function `plotvib` of this Toolbox (see Window 3.8). For instance, both commands `plot(t, x(2,:))` and `plotvib(x,2)` plot for the time history of the displacement $x_2(t)$.

Note:
Before function `ctrmdof2` can be used, function `clpmdof2` must be executed.

With the input arguments `tf` and `npts` chosen for function `clpmdof1` or `clpmdof2`, the step size h of the Runge-Kutta algorithm in Eq. (11.137) is given by

$$h = \frac{\texttt{tf}}{\texttt{npts} - 1}$$

For accuracy in numerical simulation, it is suggested that $h \leq \pi/(3\omega_n)$, where ω_n is the highest natural frequency of the mechanical system.

EXAMPLE 6.3

Consider the position control system in Example 6.2. Let the reference input $r(t)$ be a unit step function. Let the initial conditions of the mechanical system be

$$\{x(0)\} = \begin{pmatrix} -0.5 \\ 0 \end{pmatrix}, \quad \{\dot{x}(0)\} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The command

```
>> ctrmdof2([1 1],[-0.5 0; 0 0]',20,10001)
```

plots the displacements of the controlled mechanical system (Fig. 11.6.5), the error signal $e(t) = r(t) - y_s(t)$ (Fig. 11.6.6), and the control force $u(t)$ (not shown here). Obviously, this noncolocated control system is not efficient. The reader may design a better control law.

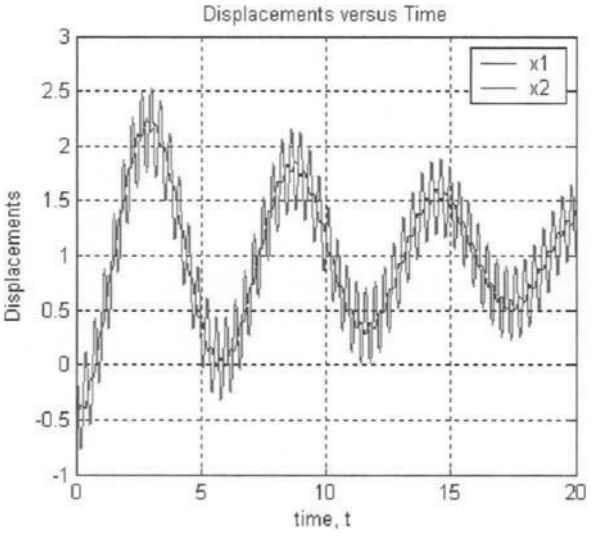


FIGURE 11.6.5

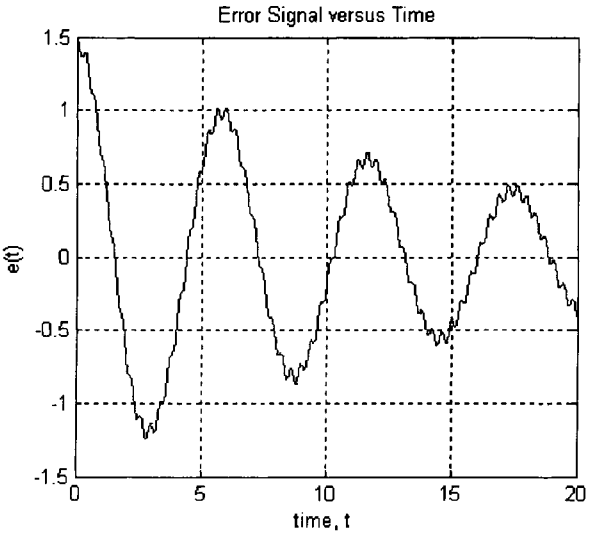


FIGURE 11.6.6

11.7 Quick Solution Guide

Governing Equations of M-DOF Systems

Equation of motion:

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \{F(t)\}, \quad t \geq 0$$

Initial conditions:

$$\{x(0)\} = \{x_0\}, \quad \{\dot{x}(0)\} = \{v_0\}$$

$[M]$ – mass matrix; $[C]$ – damping matrix; $[K]$ – stiffness matrix

$\{x(t)\}$ – vector of displacements; $\{F(t)\}$ – vector of external forces

Problems to Solve

(a) Eigenvalue problem (modes of vibration)

$$\left(\lambda^2 [M] + \lambda [C] + [K]\right)\{u\} = 0$$

(b) Free vibration

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = 0, \quad t \geq 0$$
$$\{x(0)\} = \{x_0\}, \quad \{\dot{x}(0)\} = \{v_0\}$$

(c) Forced vibration

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \{B_f\} f(t), \quad t \geq 0$$
$$\{x(0)\} = 0, \quad \{\dot{x}(0)\} = 0$$

(d) Frequency response

$$[M]\{\ddot{x}(t)\} + [C]\{\dot{x}(t)\} + [K]\{x(t)\} = \{B_f\} e^{j\omega t}$$

(e) Design of dynamic vibration absorbers

(f) Design of feedback controllers for vibration control and position control

MATLAB Functions

This chapter has a set (toolbox) of MATLAB functions for vibration analysis and feedback control of multiple-degree-of-freedom mechanical systems; see the table below. Refer to the corresponding window or section for the utility of each function.

	System Setup	
`setmdof`	Input matrices $[M]$, $[C]$, and $[K]$, and compute eigensolutions	Window 2.1
	Modes of Vibration	
`animode`	Animate vibration in a particular mode	Window 2.3
`chareq`	Obtain characteristic equation $\det\left(s^2[M] + s[C] + [K]\right) = 0$	Section 11.2.3
`geteig`	Output eigensolutions computed by function `setmdof`	Window 2.2

(continued)

Time Response		
`freeMD`	Obtain free response by modal analysis	Window 3.1
`forcedMD`	Obtain forced response by modal analysis	Window 3.2
`freeLT`	Obtain free response by inverse Laplace transform	Window 3.3
`forcedLT`	Obtain forced response by inverse Laplace transform	Window 3.4
`freeRK`	Compute free response by Runge-Kutta algorithm	Window 3.6
`forcedRK`	Compute forced response by Runge-Kutta algorithm	Window 3.7
`respCV`	Compute system response based on analyical expressions obtained by `freeMD`, `forcedMD`, `freeLT`, and `forcedLT`	Window 3.5
`plotvib`	Plot time response obtained by functions `freeRK`, `forcedRK`, `respCV`, `ctrmdof1`, and `ctrmdof2`	Window 3.8
Frequency Response		
`frfun`	Obtain frequency response (magnitude and phase)	Window 3.9
`plotfr`	Plot frequency response	Window 3.10
Dynamic Vibration Absorption		
`absorb`	Characterize the dynamic behaviors of a vibrating system combined with an undamped vibration absorber	Window 4.1
`absorbdmp`	Plot the steady-state response of a vibrating system combined with a damped vibration absorber	Window 4.2
`absorbmsratio`	Obtain a range of mass ratio for vibration absorber design	Window 4.3
`absorbopt`	Optimally tune a damped absorber	Window 4.4
Transfer Function and Green's Function		
`getTF`	Obtain transfer function $H(s) = \left(s^2[M] + s[C] + [K]\right)^{-1} [B_f]$	Window 5.1
`getGF`	Obtain Green's function $G(t) = \mathcal{L}^{-1}[H(s)]\ [B_f]$	Window 5.2
`pzTF`	Compute transfer function poles and zeros	Window 5.3
Vibration Control System		
`ssmdof1`	Obtain a state representation for vibration control of a mechanical system	Window 6.1
`clpmdof1`	Obtain the closed-loop state equation for vibration control	Window 6.2
`ctrmdof1`	Plot the response of a vibration control system	Window 6.4
Position Control System		
`ssmdof2`	Obtain a state representation for position control of a mechanical system	Window 6.1
`clpmdof2`	Obtain the closed-loop state equation for position control	Window 6.3
`ctrmdof2`	Plot the response of a position control system	Window 6.5
Utilities		
`systinfo`	Display information on system parameters and eigensolutions	Section 11.2.3
`TBdemo`	Demonstrate how the Toolbox works and what it can do	Section 11.1
`TBinfo`	Show the solution procedure of the Toolbox	Section 11.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 11.1

The following is a list of MATLAB functions whose utility requires the MATLAB Control System Toolbox:

`forcedLT, forcedMD, freeLT, freeMD, getGF, getTF, pzTF.`

Solution Procedure

To compute and plot the dynamic response of an M-DOF system, follow the three steps below. All commands are executed in the MATLAB command window.

Step 1. Input system parameters and compute eigensolutions

```
>> setmdof(M, K, Dmp_Spec)
```

Step 2. Compute system response

To animate modes of vibration, type

```
>> animode(mode_no)
```

To obtain free response, type

```
>> y = freeRK(x0,v0);   % by Runge-Kutta algorithm
>> y = freeLT(x0,v0);   % by Laplace transform
>> y = freeMD(x0,v0);   % by modal analysis
```

To obtain forced time response, type

```
>> y = forcedRK(Load_Spec);   % by Runge-Kutta algorithm
>> y = forcedLT(Load_Spec);   % by Laplace transform
>> y = forcedMD(Load_Spec);   % by modal analysis
```

To compute frequency response to a harmonic excitation, type

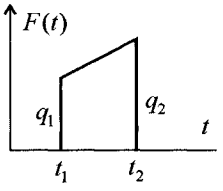
```
>> y = frfun(Bf,jx);
```

Step 3. Plot computed response

Use `plotvib(y)` to plot time-domain response where y is obtained by `freeRK`, `forcedRK`, or `respCV`.

Use `plotfr(y)` to plot frequency response where y is obtained by `frfun`

Specification of Vector `Load_Spec` for Function `forcedRK` (Same as Table 11.3.3)

External Force	$f(t)$	`Load_Spec`
Impulse force	$I_0\delta(t)$ $I_0\delta(t-T_d)$	[0 I_0] [0 I_0 T_d]
Step force	F_0 $F_0u(t-T_d)$	[1 F_0] [1 F_0 T_d]
Ramp force	$a_0\,t$ $a_0\,(t-T_d)\,u(t-T_d)$	[2 a_0] [2 a_0 T_d]
Exponential force	$A\,e^{bt}$ $A\,e^{b(t-T_d)}\,u(t-T_d)$	[3 A b] [3 A b T_d]
	$A\sin(\omega t+\phi_0)$	[4 A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
Sinusoidal force	$A\sin(\omega(t-T_d)+\phi_0)\;u(t-T_d)$	[4 A ω ϕ_0 T_d] ω in rad/s, ϕ_0 in degrees
Pulse force		[5 t_1 q_1 t_2 q_2]

Specification of Vector Load_Spec for forcedMD and forcedLT (Same as Table 11.3.2)

External Force	$f(t)$	Load_Spec
Impulse load	$I_0\delta(t)$	$[0\ I_0]$
Step load	F_0	$[1\ F_0]$
Ramp load	$a_0 t$	$[2\ a_0]$
Exponential load	$A e^{bt}$	$[3\ A\ b]$
Sinusoidal load	$A\sin(\omega t + \phi_0)$	$[4\ A\ \omega\ \phi_0]$ ω in rad/s, ϕ_0 in degrees
User-specified load	$f(t) = \mathscr{L}^{-1}[F(s)]$ where $\mathscr{L}^{-1}$ is the inverse Laplace operator, $F(s) = \dfrac{b_p s^p + b_{p-1}s^{p-1} + \cdots + b_1 s + b_0}{a_p s^p + a_{p-1}s^{p-1} + \cdots + a_1 s + a_0}$ and $a_p \neq 0$.	$\begin{bmatrix} b_p & b_{p-1} & \cdots & b_1 & b_0 \\ a_p & a_{p-1} & \cdots & a_1 & a_0 \end{bmatrix}$

11.8 References

References on Vibrations

1. Balachandran B, Magrab E B 2003 *Vibrations*, Brooks/Cole Publishing Co.: New York.
2. Den Hartog J P 1985 *Mechanical Vibrations*, Dover Publications: New York.
3. Inman D J 2000 *Engineering Vibrations*, 2nd edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
4. Meirovitch L 1967 *Analytical Methods in Vibrations*, Macmillan: New York.
5. Rao S S 2003 *Mechanical Vibrations*, 4th edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
6. Timoshenko S 1990 *Vibration Problems in Engineering*, 5th edition, John Wiley & Sons.
7. Caughey T K, O'Kelley M E J 1965 "Classical Normal Modes in Damped Linear Dynamic Systems," *ASME Journal of Applied Mechanics*, Vol. 49: pp. 867–870.

References on Feedback Control

8. Chen C-T 1998 *Linear System Theory and Design*, 3rd edition, Oxford University Press: New York.
9. Dorf R C, Bishop R H 2000 *Modern Control Systems*, 9th edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
10. Kuo B C, Golnaraghi F 2002 Automatic Control Systems, 8th edition, Wiley Text Books.
11. Nise N S 2000 Control Systems Engineering, 3rd edition, Wiley Text Books.
12. Ogata K 2001 *Modern Control Engineering*, 4th edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.

PART IV
Structural Dynamics

This Page Intentionally Left Blank

12 Dynamics and Control of Euler-Bernoulli Beams

Inside
- Getting Started
- System Description
- Dynamic Response
- Feedback Control
- Dynamics and Control of Nonuniform Beams
- Quick Solution Guide
- References

12.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for dynamic analysis and feedback control of Euler-Bernoulli beams.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on the computer
- The MATLAB Control System Toolbox installed on the computer

(Without the MATLAB Control System Toolbox, some functions from the Toolbox of this chapter cannot be used; see Section 12.6.)

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

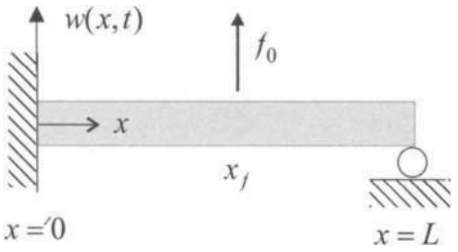


FIGURE 12.1.1

(iii) Test the Toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

Quick Tutorial

To show how to use the Toolbox, consider a clamped-pinned beam in Fig. 12.1.1, with linear density $\rho = 0.35$, bending stiffness $EI = 12$, and length $L = 2$. A pointwise step (constant) force $f_0 = 3.5$ is applied to the beam at $x_f = 0.7$. Assume that the beam is initially at rest. To compute the forced vibration of the beam at $x = 0.82$ for $0 \le t \le 2.0$, type the following commands in the MATLAB command window:

```
>> setbeam(0.35, 12, 2, [2 1])
>> y = forcedbeam([1 0.7 3.5], 0.82, 2);
>> plotbeam(y)
```

where >> is the prompt in the MATLAB command window. This yields a time history of the beam displacement $w(0.82, t)$ in Fig. 12.1.2 and the first eight natural frequencies of the beam as follows

```
First 8 Eigenvalues
-------------------
 wn, natural frequencies in rad/sec
 fn = wn/(2*pi), natural frequencies in Hertz

    wn        fn
 22.5699    3.5921
 73.1411   11.6408
152.6030   24.2875
260.9602   41.5331
398.2126   63.3775
564.3602   89.8207
759.4031  120.8628
983.3412  156.5036
```

In the above simulation, three functions from the Toolbox have been called:

(a) Function `setmbeam` that inputs system parameters, sets up boundary conditions, and computes the natural frequencies and mode shapes of the beam;

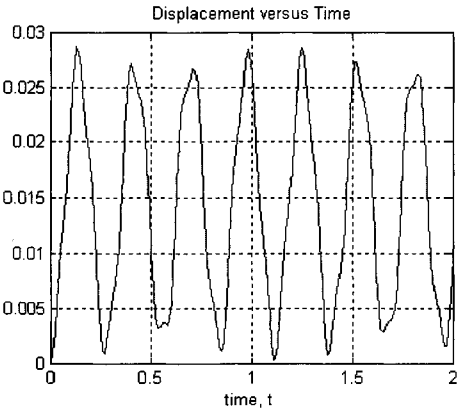


FIGURE 12.1.2

(b) Function `forcedbeam` that computes the forced response of the beam subject to the step force; and
(c) Function `plotbeam` that plots the computed beam displacement versus time.

These functions, along with others from the Toolbox, will be described in the subsequent sections.

For a quick solution, the user may refer directly to the Quick Solution Guide (Section 12.6), or get the information on the Toolbox by typing **`TBinfo`** in the MATLAB command window. To run the examples contained in this chapter, type **`RunEx`** in the MATLAB command window. To better understand how the Toolbox works, the user is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 12.7.

12.2 System Description

In this section, the governing equations and eigenvalue problem of uniform Euler-Bernoulli beams are presented; the related MATLAB functions from the Toolbox are introduced. (Nonuniform beams are covered in Section 12.5.)

12.2.1 Governing Equation

The transverse displacement $w(x,t)$ of a uniform Euler-Bernoulli beam shown in Fig. 12.2.1 is governed by the partial differential equation (Reference 6)

$$\rho \frac{\partial^2}{\partial t^2} w(x,t) + d_v \frac{\partial}{\partial t} w(x,t) + EI \frac{\partial^4}{\partial x^4} w(x,t) = F(x,t), \quad 0 \le x \le L \tag{12.1}$$

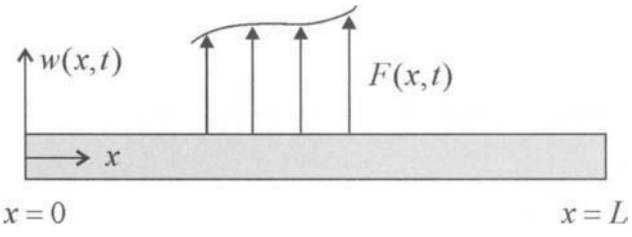


FIGURE 12.2.1 Schematic of a uniform Euler-Bernoulli beam.

with the boundary conditions

$$B_{Lj}[w(x,t)]_{x=0} = 0, \quad B_{Rj}[w(x,t)]_{x=L} = 0, \quad j = 1,2 \tag{12.2}$$

and the initial conditions

$$w(x,0) = a_0(x), \quad \frac{\partial}{\partial t}w(x,0) = b_0(x), \quad 0 \le x \le L \tag{12.3}$$

where ρ, EI, and L are the linear density (mass per unit length), bending stiffness, and length of the beam, respectively, d_v is a viscous damping coefficient, $F(x,t)$ is an external force, $a_0(x)$ and $b_0(x)$ are the spatial distributions of initial displacement and velocity of the beam, and B_{Lj} and B_{Rj} are spatial differential operators. The rotation, bending moment, and shear force of the beam are

$$\theta(x,t) = \frac{\partial}{\partial x}w(x,t), \quad M(x,t) = EI\frac{\partial^2}{\partial x^2}w(x,t), \quad Q(x,t) = EI\frac{\partial^3}{\partial x^3}w(x,t) \tag{12.4}$$

Seven types of boundary conditions are presented in Table 12.2.1, of which the first four (B1 to B4) are *classical boundary conditions* and the remaining ones *nonclassical*. For boundary conditions involved with end inertia (lumped mass or rigid disk), refer to Table 12.5.1 of Section 12.5.

Equations (12.1) to (12.3) form a boundary-initial value problem. A fundamental problem in dynamic analysis of the Euler-Bernoulli beam is to solve these equations for the dynamic response (displacement, rotation, bending moment, and shear force) of the beam.

12.2.2 Eigenvalue Problem

Eigenvalue problems play an essential role in dynamic analysis of flexible structures. In this section, the eigenvalue problem of the Euler-Bernoulli beam is formulated. To this end, assume that the displacement of the beam is

$$w(x,t) = u(x)e^{j\omega t}, \quad j = \sqrt{-1} \tag{12.5}$$

Substituting Eq. (12.5) into Eqs. (12.1) and (12.2) with $F(x,t) = 0$ yields the eigen equation

$$\rho\omega^2 u(x) = EI\frac{d^4}{dx^4}u(x), \quad 0 \le x \le L \tag{12.6}$$

and the boundary conditions

$$B_{Li}[u(x)]_{x=0} = 0, \quad B_{Ri}[u(x)]_{x=L} = 0, \quad i = 1,2 \tag{12.7}$$

where ω is an eigenvalue and $u(x)$ is an eigenfunction. Physically, ω is a natural frequency of the beam, while $u(x)$ is the associate mode shape. Equation (12.6) has an infinite number of solutions: $(\omega_k,\ u_k(x))$, $k = 1,2,3,\ldots$ The natural frequencies are nonnegative, and can be arranged in an ascending order

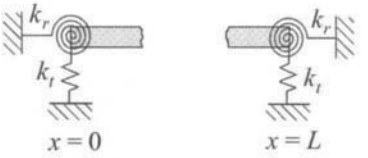
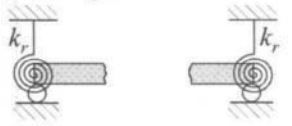
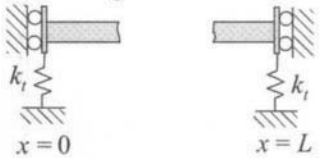
$$0 \le \omega_1 \le \omega_2 < \omega_3 < \cdots \tag{12.8}$$

The eigenfunctions can be scaled to satisfy the orthonormal conditions

$$\rho\int_0^L u_j(x)u_k(x)\,dx = \delta_{jk}, \quad EI\int_0^L u_j(x)\frac{d^4}{dx^4}u_k(x)\,dx = \delta_{jk}\omega_k^2 \tag{12.9}$$

where the delta function δ_{jk} is one for $j = k$ and zero for $j \ne k$.

TABLE 12.2.1 Boundary Conditions of Beams in Transverse Vibration
(k_r - torsional spring coefficient; k_t - translational spring coefficient)

Boundary Type	*Boundary Conditions*
(B1) Pinned or hinged end	Left end: $w(0,t)=0, \quad -EI\dfrac{\partial^2 w(0,t)}{\partial x^2}=0$ Right end: $w(L,t)=0, \quad EI\dfrac{\partial^2 w(L,t)}{\partial x^2}=0$
(B2) Clamped or fixed end	Left end: $w(0,t)=0, \quad \dfrac{\partial w(0,t)}{\partial x}=0$ Right end: $w(L,t)=0, \quad \dfrac{\partial w(L,t)}{\partial x}=0$
(B3) Free end	Left end: $-EI\dfrac{\partial^2 w(0,t)}{\partial x^2}=0, \quad EI\dfrac{\partial^3 w(0,t)}{\partial x^3}=0$ Right end: $EI\dfrac{\partial^2 w(L,t)}{\partial x^2}=0, \quad -EI\dfrac{\partial^3 w(L,t)}{\partial x^3}=0$
(B4) Sliding or guided end	Left end: $\dfrac{\partial w(0,t)}{\partial x}=0, \quad EI\dfrac{\partial^3 w(0,t)}{\partial x^3}=0$ Right end: $\dfrac{\partial w(L,t)}{\partial x}=0, \quad -EI\dfrac{\partial^3 w(L,t)}{\partial x^3}=0$
(B5) Elastic-support 	Left end: $k_r\dfrac{\partial w(0,t)}{\partial x}-EI\dfrac{\partial^2 w(0,t)}{\partial x^2}=0, \quad k_t w(0,t)+EI\dfrac{\partial^3 w(0,t)}{\partial x^3}=0$ Right end: $k_r\dfrac{\partial w(L,t)}{\partial x}+EI\dfrac{\partial^2 w(L,t)}{\partial x^2}=0, \quad k_t w(L,t)-EI\dfrac{\partial^3 w(L,t)}{\partial x^3}=0$
(B6) Hinged-elastic end 	Left end: $w(0,t)=0, \quad k_r\dfrac{\partial w(0,t)}{\partial x}-EI\dfrac{\partial^2 w(0,t)}{\partial x^2}=0$ Right end: $w(L,t)=0, \quad k_r\dfrac{\partial w(L,t)}{\partial x}+EI\dfrac{\partial^2 w(L,t)}{\partial x^2}=0$
(B7) Sliding-elastic end 	Left end: $\dfrac{\partial w(0,t)}{\partial x}=0, \quad k_t w(0,t)+EI\dfrac{\partial^3 w(0,t)}{\partial x^3}=0$ Right end: $\dfrac{\partial w(L,t)}{\partial x}=0, \quad k_t w(L,t)-EI\dfrac{\partial^3 w(L,t)}{\partial x^3}=0$

An eigenfunction associated with a nonzero eigenvalue is called a *flexible mode*; an eigenfunction associated with a zero eigenvalue is called a *rigid-body mode*. For a nonzero eigenvalue ω, the corresponding eigenfunction can be written as

$$u(x) = a_1 \cos \beta x + a_2 \sin \beta x + a_3 \cosh \beta x + a_4 \sinh \beta x \tag{12.10}$$

where the parameters β and ω are related by

$$\beta^4 = \omega^2 \frac{\rho}{EI} \tag{12.11}$$

For a zero eigenvalue, the corresponding eigenfunction is

$$u(x) = r_1 + r_2 x \tag{12.12}$$

where r_1 and r_2 are constants. For instance, a free-free beam has two zero eigenvalues representing rigid-body modes in rotational motion and translational motion, respectively.

To solve the eigenvalue problem, substitute Eq. (12.10) into the boundary conditions (12.7), yielding the homogenous equation

$$[A(\mu)]\{a\} = 0 \tag{12.13}$$

where μ is the nondimensional eigenvalue

$$\mu = \beta L = \sqrt[4]{\frac{\rho}{EI}} \sqrt{\omega} L \tag{12.14}$$

$\{a\} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \end{pmatrix}^T$, and $[A(\mu)]$ is a matrix whose elements are transcendental functions of μ. For nonzero solution of $\{a\}$, the determinant of $[A(\mu)]$ must be zero, which renders the characteristic equation

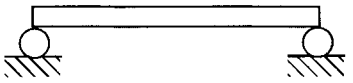
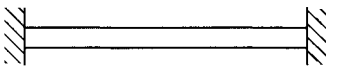
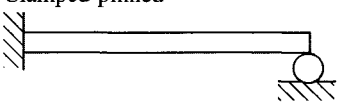
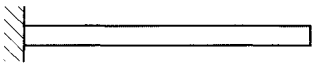
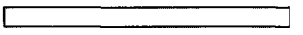
$$\Delta(\mu) = \det [A(\mu)] = 0 \tag{12.15}$$

The characteristic equation $\Delta(\mu)$ has an infinite number of roots μ_k, $k = 1, 2, \ldots$, which are related to the natural frequencies of the beam by

$$\omega_k = \left(\frac{\mu_k}{L}\right)^2 \sqrt{\frac{EI}{\rho}} \tag{12.16}$$

For a known eigenvalue μ_k (or ω_k), vector $\{a\}$ is a nontrivial solution of $[A(\mu_k)]\{a\} = 0$; the modal shape $u_k(x)$ can be evaluated by Eq. (12.10) or Eq. (12.12). Table 12.2.2 gives the eigensolutions for certain beam configurations.

TABLE 12.2.2 Eigensolutions of Euler-Bernoulli Beam with Certain Boundary Conditions

Boundary Conditions and Characteristic Equation	$\mu_k = \beta_k L$	*Eigenfunction* $u_k(x)$*
Pinned-pinned (simply supported)  $\sin \mu = 0$	3.141597 6.283185 9.424778 12.56637 $\mu_k = k\pi$ for $k \geq 1$	$\sin \beta_k x$
Clamped-clamped  $\cos \mu \cdot \cosh \mu = 1$	4.730041 7.853205 10.99561 14.13717 $\mu_k \approx \frac{1}{2}(2k+1)\pi$ for $k \geq 5$	$\cosh \beta_k x - \cos \beta_k x$ $-\alpha_k(\sinh \beta_k x - \sin \beta_k x)$ $\alpha_k = \frac{\cosh \mu_k - \cos \mu_k}{\sinh \mu_k - \sin \mu_k}$
Clamped-pinned  $\tan \mu = \tanh \mu$	3.926602 7.068583 10.21018 13.35177 $\mu_k \approx \frac{1}{4}(4k+1)\pi$ for $k \geq 3$	$\cosh \beta_k x - \cos \beta_k x$ $-\alpha_k(\sinh \beta_k x - \sin \beta_k x)$ $\alpha_k = \frac{\cosh \mu_k - \cos \mu_k}{\sinh \mu_k - \sin \mu_k}$
Clamped-free (cantilever)  $\cos \mu \cdot \cosh \mu = 1$	1.875104 4.694091 7.854757 10.99554 $\mu_k \approx \frac{1}{2}(2k-1)\pi$ for $k \geq 5$	$\cosh \beta_k x - \cos \beta_k x$ $-\alpha_k(\sinh \beta_k x - \sin \beta_k x)$ $\alpha_k = \frac{\cosh \mu_k + \cos \mu_k}{\sinh \mu_k + \sin \mu_k}$
Free-free  $\mu^2 = 0$ and $\cos \mu \cdot \cosh \mu = 1$	Two zero eigenvalues: 0, 0 Nonzero eigenvalues: 4.730041 7.853205 10.99561 14.13717 $\mu_k \approx \frac{1}{2}(2k+1)\pi$ for $k \geq 5$	Two rigid-body modes: $u(x) =$ constant $u(x) = x - L/2$ Flexible modes: $\cosh \beta_k x + \cos \beta_k x$ $-\alpha_k(\sinh \beta_k x + \sin \beta_k x)$ $\alpha_k = \frac{\cosh \mu_k - \cos \mu_k}{\sinh \mu_k - \sin \mu_k}$

*The eigenfunctions listed herein are not normalized.

As an example, consider a beam with clamped-pinned ends, whose eigenfunctions satisfy the boundary conditions

$$u(0) = 0, \quad u'(0) = 0; \quad u(L) = 0, \quad u''(L) = 0$$

where $u' = du/dx$. Substituting Eq. (12.10) into the boundary conditions gives

$$[A(\mu)]\{a\} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & \mu & 0 & \mu \\ \cos \mu & \sin \mu & \cosh \mu & \sinh \mu \\ -\mu^2 \cos \mu & -\mu^2 \sin \mu & \mu^2 \cosh \mu & \mu^2 \sinh \mu \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = 0$$

which, by Eq. (12.15), leads to the characteristic equation

$$\tan \mu = \tanh \mu$$

The first two characteristic roots are $\mu_1 = 3.926602$ and $\mu_2 = 7.068583$. The mode shapes are

$$u_k(x) = A_k \left\{ \cosh \beta_k x - \cos \beta_k x \; - \frac{\cosh \mu_k - \cos \mu_k}{\sinh \mu_k - \sin \mu_k} (\sinh \beta_k x - \sin \beta_k x) \right\}$$

where A_k is a nonzero constant.

12.2.3 System Setup by the Toolbox

The Toolbox of this chapter provides a MATLAB function **setbeam** for specifying the parameters and boundary conditions of an Euler-Bernoulli beam, and for computing the beam eigensolutions; see Window 2.1.

Window 2.1. Function setbeam

MATLAB Function: **setbeam**

Purpose:
To specify the parameters and boundary conditions of a Euler-Bernoulli beam, and to compute the natural frequencies and mode shapes of the beam.

Synopsis:

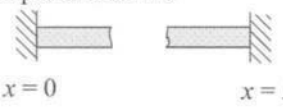
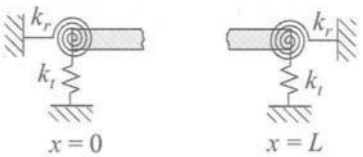
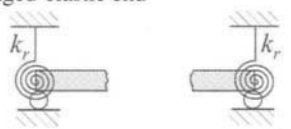
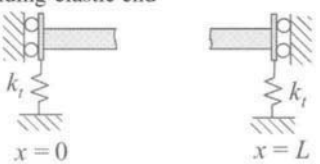
```
setbeam(rou, EI, L, BC_Spec)
setbeam(rou, EI, L, BC_Spec, Dmp_Spec)
setbeam(rou, EI, L, BC_Spec, Dmp_Spec, n_mode)
```

Description:
`setbeam(rou, EI, L, BC_Spec)` undertakes the following tasks:
(a) it inputs linear density `rou`, bending stiffness `EI`, and beam length `L`; (b) it sets up boundary conditions by a vector `BC_Spec` according to Table 12.2.3; and (c) it computes beam eigenvalues (natural frequencies) and normalized eigenfunctions (mode shapes).

`setbeam(rou, EI, L, BC_Spec, Dmp_Spec)` introduces damping in the beam with the following options:
(a) Viscous damping: `Dmp_Spec` $= d_v$, where d_v is a viscous damping coefficient;
(b) Modal damping: `Dmp_Spec = [1 zeta_1, zeta_2 ... zeta_m]`, where `zeta_k` is the modal damping ratio of the kth mode. All damping ratios fall between 0 and 1. If $m <$ `n_mode`, where n_mode is the total number of modes considered, modes $m + 1, m + 2, \ldots,$ `n_mode` have the same damping ratio as `zeta_m`. For instance, `Dmp_Spec = [2 0.1 0.05]` specifies a damping ratio of 0.1 for the first mode, and a damping ratio of 0.05 for the rest modes;
(c) No damping: `Dmp_Spec = [ ] or 0`.

`setbeam(rou, EI, L, BC_Spec, Dmp_Spec, n_mode)` selects the total number `n_mode` of modes considered for simulation. By default, `n_mode` = 8.

TABLE 12.2.3 Specification of Boundary Conditions

Boundary Condition (k_r - torsional spring coefficient; k_t - translational spring coefficient)	`BC_Spec = [BC_Left BC_Right]`	
	Left Boundary ($x = 0$) `BC_Left`	*Right Boundary ($x = L$)* `BC_Right`
(B1) Pinned or hinged end	[1]	[1]
(B2) Clamped or fixed end 	[2]	[2]
(B3) Free end	[3]	[3]
(B4) Sliding or guided end	[4]	[4]
(B5) Elastic-support 	[5 k_r k_t]	[5 k_r k_t]
(B6) Hinged-elastic end 	[6 k_r]	[6 k_r]
(B7) Sliding-elastic end 	[7 k_t]	[7 k_t]

Note 1. Function `setbeam` must be executed before any other functions of dynamic analysis and feedback control from the Toolbox can be used. However, `setbeam` only needs to be called once for the same beam in different cases of initial and external disturbances.

Note 2. Function `setbeam` presents beam eigenvalues by the following four parameters:

ω_k, natural frequencies in rad/sec

$f_k = \omega_k/2\pi$, natural frequencies in Hertz

$$\beta_k = \sqrt[4]{\frac{\rho}{EI}}\ \sqrt{\omega_k}$$

$\mu_k = \beta_k L$, nondimensional eigenvalues

Note 3. Function `setbeam` presents normalized eigenfunctions in the following form:

Flexible mode with a nonzero eigenvalue:

$$u_k(x) = a_1 e^{-\beta_k(L-x)} + a_2 e^{-\beta_k x} + a_3 \cos \beta_k x + a_4 \sin \beta_k x \tag{12.17a}$$

Rigid-body mode with a zero eigenvalue:

$$u_k(x) = r_1 + r_2 x \tag{12.17b}$$

where a_i's and r_i's are such that $\rho \int_0^L u_k^2(x)\, dx = 1$. Equation (12.17a) is equivalent to Eq. (12.10).

Note 4. To get the information of beam parameters, boundary conditions, and eigensolutions that are provided by `setbeam`, function `systinfo` can be called at any time. This is done by typing the following in the MATLAB command window:

```
>> systinfo
```

Note 5. For beams with classical boundary conditions (B1 to B4 in Table 12.2.1), the nondimensional eigenvalues μ_k are independent of beam parameters, ρ, EI, and L. In other words, once μ_k's are known, the natural frequencies of a beam of arbitrary parameter can be computed by Eq. (12.16). See Table 12.2.2 for some examples.

EXAMPLE 2.1

For the beam configurations given in Fig. 12.2.2, the boundary specification vector is formed according to Table 12.2.3:
Case (a) Pinned-pinned beam (B1-B1):

```
BC_Spec = [1 1]
```

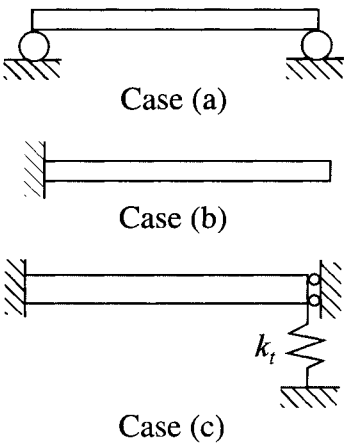


FIGURE 12.2.2

Case (b) Clamped-free beam (B2-B3):

```
BC_Spec = [2 3]
```

Case (c) Clamped-sliding-elastic beam (B2-B7):

```
BC_Spec = [1 7 k_t]
```

EXAMPLE 2.2

Consider a pinned-free beam of parameters $\rho = 0.5$, $EI = 40$, and $L = 2$. Let the beam be undamped. The command

```
>> setbeam(0.5, 40, 2, [1 3])
```

gives the first eight natural frequencies of the beam (in rad/s)

```
0
34.4762
111.725
233.105
398.623
608.280
862.074
1160.01
```

plots the first four mode shapes in Fig. 12.2.3, and shows the coefficients of the normalized eigenfunctions as follows

```
Normalized Eigenfunctions
-------------------------
Rigid-body Modes: u(x)=r1+r2*x
```

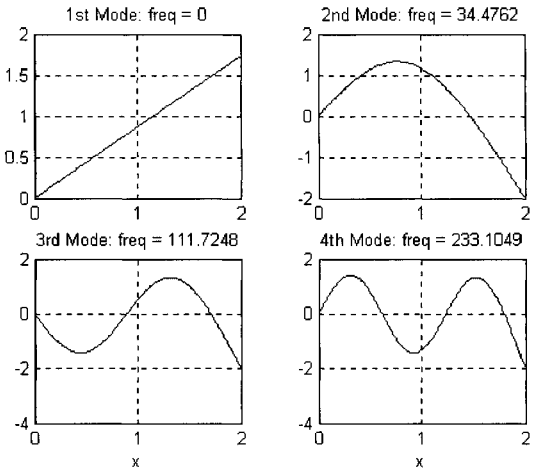


FIGURE 12.2.3

```
Mode No. r1 r2
1.0000 0 0.8660

Flexible Modes: u(x)=a1*exp(-beta*(L-x))+a2*exp(-beta*x)+
  a3*cos(beta*x)+a4*sin(beta*x)
Mode No.  beta     a1     a2    a3    a4
1.0000  1.9633 -1.0004 0.0197   0  1.4148
2.0000  3.5343 -1.0000 0.0009   0 -1.4142
3.0000  5.1051 -1.0000 0.0000   0  1.4142
4.0000  6.6759 -1.0000      0   0 -1.4142
5.0000  8.2467 -1.0000      0   0  1.4142
6.0000  9.8175 -1.0000      0   0 -1.4142
7.0000 11.3883 -1.0000      0   0  1.4142
```

Here the zero eigenvalue represents a rigid-body mode in rotational motion about the pinned end.

Besides `setbeam`, functions `eigenbeam`, `plotmsh`, and `nodalpts` of the Toolbox are useful for computing and displaying beam eigensolutions; see Windows 2.2 to 2.4.

Window 2.2. Function `eigenbeam`

MATLAB Function: **`eigenbeam`**

Purpose:
To determine the eigenvalues and normalized eigenfunctions of a beam.

Synopsis:

```
[y, msh] = eigenbeam(n)
```

Description:
`[y, msh] = eigenbeam(n)` returns the first `n` eigenvalues $(\omega_1, \omega_2, \ldots, \omega_n)$ of a beam in a vector `y` and the coefficients of normalized eigenfunctions in a five-by-n matrix `msh`. The kth column of `msh` describes the kth eigenfunction $u_k(x)$ as follows:

(i) if $u_k(x)$ is a flexible mode

`msh(1,k)` = β_k, `msh(2,k)` = a_1, `msh(3,k)` = a_2, `msh(4,k)` = a_3, `msh(5,k)` = a_4;

(ii) if $u_k(x)$ is a rigid-body mode

`msh(1,k) = 0, msh(2,k)` = r_1, `msh(3,k)` = r_2, `msh(4,k) = 0, msh(5,k) = 0.`

where β_k, a_i, and r_j are the parameters given in Eqs. (12.17).

Window 2.3. Function `plotmsh`

MATLAB Function: **`plotmsh`**

Purpose:
To plot spatial distributions of modal displacement and internal forces of a beam

Synopsis:

```
plotmsh(n)
plotmsh(n, opt, npts)
[y, x] = plotmsh(n, opt, npts)
```

Description:
`plotmsh(n)` plots the nth mode shape $u_n(x)$ of a beam over its spatial region $[0, L]$.

`plotmsh(n, opt, npts)` plots the spatial distribution of nth mode with the options:

`opt` = 1	plot modal displacement $u_n(x)$ versus x
`opt` = 2	plot modal rotation $u_n'(x)$ versus x
`opt` = 3	plot modal bending moment $M(x) = EIu_n''(x)$ versus x
`opt` = 4	plot modal shear force $Q(x) = EIu_n'''(x)$ versus x

Here `npts` is the number of equally-spaced points in the region $[0, L]$ that are selected for computation. By default, `opt` = 1 and `npts` = 201. Either of `opt` and `npts` can be replaced by `[ ]` as a placeholder for the default value.

`[y, x] = plotmsh(n, opt, npts)` returns the modal distribution in a vector `y` and spatial points of computation in a vector `x`. With `x` and `y`, the beam modal distribution can be plotted by the MATLAB function `plot(x, y)`.

Window 2.4. Function `nodalpts`

MATLAB Function: **`nodalpts`**

Purpose:
To determine the nodal points of an eigenfunction of a beam.

Synopsis:

```
nodalpts(k)
y = nodalpts(k)
```

Description:
`nodalpts(k)` displays the nodal points of the kth mode shape $u_k(x)$ of a beam. A nodal point x_* of $u_k(x)$ is a point between 0 and L such that $u_k(x_*) = 0$.

`y = nodalpts(k)` returns the nodal points of the kth mode in a vector `y`.

Note: Function `nodalpts` is automatically called from inside of functions `setbeam` and `plotmsh`.

EXAMPLE 2.3

For the beam considered in Example 2.2, the modal bending moment for the first three modes is plotted in Fig. 12.2.4 by

```
>> setbeam(0.5, 40, 2, [1 3])
>> [y1, x] = plotmsh(1,3);
>> y2 = plotmsh(2,3);
>> y3 = plotmsh(3,3);
>> clf
>> plot(x, y1, x, y2, ':', x, y3, '-.');
>> legend('mode 1', 'mode 2', 'mode 3')
>> xlabel('x')
>> title('Modal bending moment versus x')
```

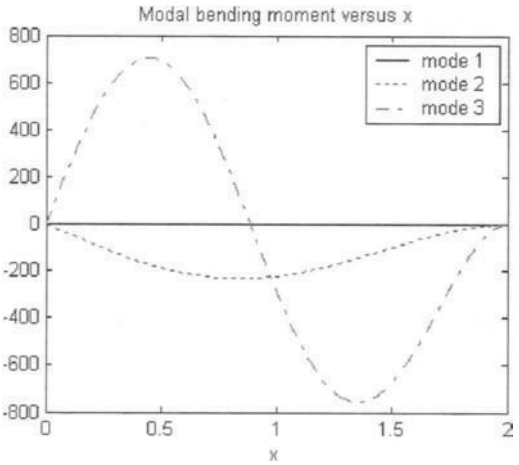


FIGURE 12.2.4

12.2.4 Animation of Modes of Vibration

The Toolbox of this chapter has a function for illustrating the vibration of a beam in a particular mode; see Window 2.5.

Window 2.5. Functions animode

MATLAB Function: **animode**

Purpose:
To animate a mode of vibration of a beam in its spatial domain $0 \leq x \leq L$.

Synopsis:

```
animode(k)
animode(k, tf, n_frames, IS_control)
F = animode(k, tf, n_frames, IS_control)
```

Window 2.5. Function animode (*Continued*)

Description:
`animode(k)` animates the kth mode of vibration of a beam.

`animode(k, tf, n_frames, IS_control)` plays the animation of the kth mode of vibration in a specified manner, where `tf` is total animation time, `n_frames` is the number of frames in the animation, and `IS_control` is an animation control parameter with the options:

`IS_control = 0`	play frames continuously
`IS_control = 1`	play frames one by one

By default, `tf` is eight times the period $(1/\omega_k)$ of the mode, `n_frames = 97`, and `IS_control = 0`. Each of `tf`, `n_frames`, and `IS_control` can be replaced by `[ ]` as a placeholder for its default value.

`F = animode(k, tf, n_frames, IS_control)` returns a set of frames of beam vibration in the kth mode in a matrix `F`, which can be played back by the MATLAB function `movie(F)`.

EXAMPLE 2.4

Consider a clamped-pinned beam with parameters $\rho = 0.4$, $EI = 2{,}400$, $L = 6$.

```
>> setbeam(0.4, 2400, 6, [2 1])
>> y = eigenbeam;
>> T = 2*pi/y(2);
>> animode(2, T, 7, 1)
```

animates the second mode of vibration of the beam. Shown in Fig. 12.2.5 are six frames of the mode shape generated by function `animode` at times $t_i = i \times h$, $i = 0, 1, \ldots, 5$, where $h = \frac{1}{6}\frac{2\pi}{\omega_2} = 0.0097407$, with $\omega_2 = 107.5073$ being the second natural frequency of the beam.

12.2.5 Distributed Transfer Function Method

In this section, the Distributed Transfer Function Method (DTFM), which is used in function `setbeam` for obtaining exact eigensolutions of Euler-Bernoulli beams, is briefly introduced. Refer to Appendix C for more information on the method. The user who is only interested in using the Toolbox may skip this section.

In the DTFM, the eigenvalue problem, Eq. (12.6), is cast in the equivalent spatial state form

$$\frac{d}{dx}\{\eta(x)\} = [F(\omega)]\{\eta(x)\} \tag{12.18}$$

subject to the boundary condition

$$[M_b]\{\eta(0)\} + [N_b]\{\eta(L)\} = 0 \tag{12.19}$$

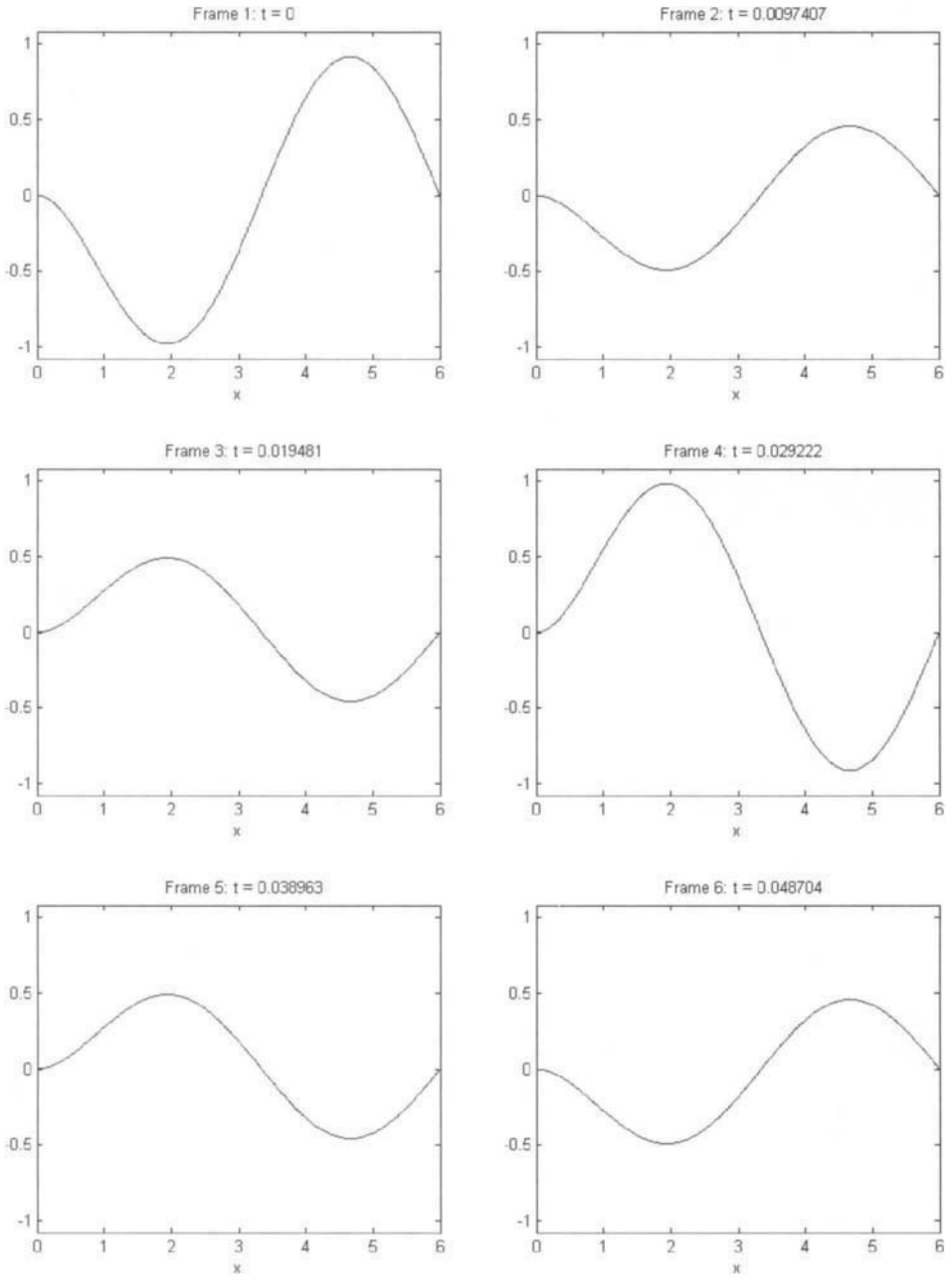


FIGURE 12.2.5 Six frames of the animated beam vibration in the second mode.

where

$$\{\eta(x)\} = \begin{pmatrix} u(x) & u'(x) & u''(x) & u'''(x) \end{pmatrix}^T$$

$$[F(\omega)] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \beta^4 & 0 & 0 & 0 \end{bmatrix}, \quad \text{with } \beta^4 = \omega^2 \frac{\rho}{EI} \tag{12.20}$$

and $[M_b]$ and $[N_b]$ are four-by-four boundary matrices. For instance, for a clamped-free (cantilever) beam, the boundary matrices are

$$[M_b] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad [N_b] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & EI & 0 \\ 0 & 0 & 0 & -EI \end{bmatrix}$$

Write the solution of Eq. (12.18) as

$$\eta(x) = e^{[F(\omega)]x}\eta(0) \tag{12.21}$$

where $e^{[F(\omega)]x}$ is the exponential of the matrix $[F(\omega)]\,x$. For the definition and properties of an exponential matrix, see Appendix A.9. Substituting Eq. (12.21) into the boundary conditions (12.19) gives the homogeneous equation

$$\left([M_b] + [N_b]\, e^{[F(\omega)]L}\right)\{\eta(0)\} = 0 \tag{12.22}$$

The characteristic equation of the beam then is

$$\Delta(\omega) \equiv \det\left([M_b] + [N_b]\, e^{[F(\omega)]L}\right) = 0 \tag{12.23}$$

where $\Delta(\omega)$ is a transcendental function with an infinite number of roots. The characteristic equation can be accurately and easily solved by standard root-searching techniques.

The eigensolutions of a beam are computed in two steps. First, solve the characteristic equation (12.23) for the natural frequencies ω_1, ω_2, ... Second, substitute ω_k into Eq. (12.22) to obtain a nonzero vector $\{\eta(0)\}$, by which the associate mode shape

$$u_k(x) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} e^{[F(\omega)]x}\{\eta(0)\} \tag{12.24}$$

12.3 Dynamic Response

In this section, the dynamic response of a uniform beam is obtained by modal expansion, which is a commonly used analytical solution technique in structural dynamics.

12.3.1 Modal Expansion

A fundamental problem in dynamic analysis of a beam is to solve the following equations:

Equation of motion:

$$\rho\frac{\partial^2}{\partial t^2}w(x,t) + d_v\frac{\partial}{\partial t}w(x,t) + EI\frac{\partial^4}{\partial x^4}w(x,t) = F(x,t), \quad 0 \le x \le L \tag{12.25}$$

Boundary conditions:

$$B_{Lj}\,[w(x,t)]_{x=0} = 0, \quad B_{Rj}\,[w(x,t)]_{x=L} = 0, \quad j = 1,2 \tag{12.26}$$

Initial conditions:

$$w(x,0) = a_0(x), \quad \frac{\partial}{\partial t} w(x,0) = b_0(x), \quad 0 \le x \le L \tag{12.27}$$

The external force in consideration has the form

$$F(x,t) = D(x) f(t) \tag{12.28}$$

where $D(x)$ describes the spatial distribution of the force.

In solution by modal expansion, the beam displacement is expressed by the series

$$w(x,t) = \sum_{k=1}^{\infty} u_k(x) q_k(t) \tag{12.29}$$

where $u_k(x)$ are the eigenfunctions of the beam, and $q_k(t)$ are unknown modal coordinates to be determined. Substituting the eigenfunction series (12.29) into Eq. (12.25) and making use of Eq. (12.6), gives

$$\sum_{k=1}^{\infty} \rho u_k(x) \left\{ \ddot{q}_k(t) + \frac{d_v}{\rho} \dot{q}_k(t) + \omega_k^2 q_k(t) \right\} = D(x) f(t) \tag{12.30}$$

Multiplying both sides of Eq. (12.30) by $u_j(x)$, integrating the resulting equation over the spatial domain $0 \le x \le L$, and making use of the orthonormal conditions (12.9), yields the differential equations for the modal coordinates

$$\ddot{q}_k(t) + 2\xi_k \omega_k \dot{q}_k(t) + \omega_k^2 q_k(t) = b_k f(t), \quad k = 1, 2, \ldots \tag{12.31}$$

where

$$b_k = \int_0^L D(x) u_k(x)\, dx \tag{12.32}$$

and the damping ratio for the kth mode is given by

$$\xi_k = \frac{d_v}{2\rho\omega_k} \tag{12.33}$$

Also, the initial conditions given in Eq. (12.27) are converted to

$$q_k(0) = \rho \int_0^L a_0(x)\, u_k(x)\, dx, \quad \dot{q}_k(0) = \rho \int_0^L b_0(x)\, u_k(x)\, dx \tag{12.34}$$

Solution of Eq. (12.31) for the modal coordinates $q_k(t)$ eventually yields the dynamic response of the beam in the eigenfunction series (12.29). This process is called *modal analysis*.

In computation, truncation of the series in terms of the first N modes has to be made, which gives the displacement and velocity of the beam as follows

$$\begin{aligned} w(x,t) &\approx w_N(x,t) = \sum_{k=1}^{N} u_k(x) q_k(t) \\ \frac{\partial}{\partial t} w(x,t) &\approx \frac{\partial}{\partial t} w_N(x,t) = \sum_{k=1}^{N} u_k(x) \dot{q}_k(t) \end{aligned} \tag{12.35}$$

Determination of $q_k(t)$ for initial and external disturbances is discussed in sequel.

12.3.2 Specification of Damping

Two types of damping are considered here: viscous damping and modal damping. It is assumed that the vibration of a beam is either undamped or underdamped. In other words, the damping ratio for each mode is such that

$$0 \leq \xi_k < 1. \tag{12.36}$$

Viscous damping (d_v) has been introduced in Eq. (12.25). By Eq. (12.33) and condition (12.36), the viscous damping coefficient must satisfy

$$0 \leq d_v < 2\rho\,\omega_{\min} \tag{12.37}$$

where $\omega_{\min}$ is the lowest nonzero natural frequency of the beam.

In dynamic analysis of a beam, modal damping is introduced in two steps: (i) Apply modal expansion to an *undamped* beam model, i.e., Eq. (12.25) with $d_v = 0$, yielding

$$\ddot{q}_k(t) + \omega_k^2 q_k(t) = b_k f(t) \tag{12.38}$$

(ii) Add a damping term $2\xi_k\omega_k\dot{q}_k(t)$ to Eq. (12.38), rendering Eq. (12.31). In this case, the damping ratio is not constrained by Eq. (12.33) as long as $0 \leq \xi_k < 1$.

Although viscous damping and modal damping have the same modal equation (12.31), they are different in two major aspects. First, viscous damping is introduced in the physical parameter space while modal damping is specified in the modal parameter (eigen) space. Second, ξ_k in the viscous damping case decreases as the mode number k increases, as indicated by Eq. (12.33), while ξ_k in the modal damping case can be arbitrarily assigned between 0 and 1.

Damping specification by the Toolbox is described in Windows 3.1 and 3.2.

Window 3.1. Damping specification vector `Dmp_Spec`

Vector: **`Dmp_Spec`** (as an argument in functions `setbeam` and `setdamp`)

Purpose:
To specify damping in a beam.

Window 3.1. Damping specification vector Dmp_Spec (*Continued*)

Description:
Vector Dmp_Spec is used to specify the damping status of a beam with the following options:
(a) Viscous damping: Dmp_Spec = d_v, where d_v is a viscous damping coefficient. The viscous damping coefficient must satisfy Eq. (12.36).
(b) Modal damping: Dmp_Spec is a vector of the form

```
Dmp_Spec = [1 zeta_1, zeta_2 ... zeta_m]
```

where zeta_j specifies the jth clamping ratio, with $0 \leq$ zeta_j < 1.
(c) No damping: Dmp_Spec = [ ] or 0.

Window 3.2. Function setdamp

MATLAB Function: **setdamp**

Purpose:
To specify or reset damping in a beam.

Synopsis:

```
setdamp(Dmp_Spec)
```

Description:
setdamp(Dmp_Spec) specifies damping in a beam, and computes and displays damping ratios for all modes selected for simulation. The damping specification vector Dmp_Spec is assigned according to Window 3.1 or 2.1.

If damping is initially set up by function setbeam, function setdamp can be called to change damping ratios for the modes in consideration or to reset damping status (no damping, viscous damping, or modal damping).

12.3.3 Free Vibration

For a beam in free vibration, its modal coordinates are governed by

$$\ddot{q}_k(t) + 2\xi_k\omega_k\dot{q}_k(t) + \omega_k^2 q_k(t) = 0, \quad k = 1, 2, \ldots \tag{12.39}$$

with the initial conditions given in Eq. (12.34). The solution of Eq. (12.39) is obtained in two cases.

Case 1. $\omega_k \neq 0$ (flexible mode):

$$q_k(t) = e^{-\xi_k\omega_k t}\left(q_k(0)\cos\omega_{dk}t + \frac{\dot{q}_k(0) + \xi_k\omega_k q_k(0)}{\omega_{dk}}\sin\omega_{dk}t\right)$$

$$\dot{q}_k(t) = -\xi_k\omega_k q_k(t) + e^{-\xi_k\omega_k t}\left[-\omega_{dk}q_k(0)\sin\omega_{dk}t + (\dot{q}_k(0) + \xi_k\omega_k q_k(0))\cos\omega_{dk}t\right] \tag{12.40}$$

with $\omega_{dk} = \omega_k\sqrt{1-\xi_k^2}$.

Case 2. $\omega_k = 0$ (rigid-body mode): If the beam has viscous damping, Eq. (12.39) reduces to

$$\ddot{q}_k(t) + (d_v/\rho)\dot{q}_k(t) = 0$$

which has the solution

$$\begin{aligned} q_k(t) &= e^{-\sigma t} q_k(0) + \frac{1}{\sigma}\left(1 - e^{-\sigma t}\right)\left(\dot{q}_k(0) + \sigma q_k(0)\right) \\ \dot{q}_k(t) &= e^{-\sigma t}\dot{q}_k(0) \end{aligned} \tag{12.41a}$$

with $\sigma = d_v/\rho$. If the beam has modal damping or no damping, Eq. (12.39) reduces to $\ddot{q}_k(t) = 0$ with the solution

$$\begin{aligned} q_k(t) &= q_k(0) + \dot{q}_k(0)\, t \\ \dot{q}_k &= \dot{q}_k(0) \end{aligned} \tag{12.41b}$$

The Toolbox of this chapter provides functions `freebeam` and `freebeam1` for computing free vibration of a beam; see Window 3.3. In using these functions, the initial values of the modal coordinates are estimated by numerical integration of Eq. (12.34)

$$q_k(0) \approx \frac{\rho L}{np}\sum_{j=1}^{np} a_0(x_j)\, u_k(x_j), \quad \dot{q}_k(0) \approx \frac{\rho L}{np}\sum_{j=1}^{np} b_0(x_j)\, u_k(x_j) \tag{12.42}$$

where $x_j = j \cdot \Delta x = j \cdot L/np, j = 1, 2, \ldots, np$, are equally-spaced points along the beam length.

Window 3.3. Functions `freebeam` and `freebeam1`

MATLAB Functions: **`freebeam, freebeam1`**

Purpose:
To compute and display free response of a beam under initial disturbances.

Synopsis:

```
freebeam(IC, xr)
[y, t] = freebeam(IC, xr, tf, npts)
freebeam1(IC, t)
[w, x] = freebeam1(IC, t, npts)
```

Description:
`freebeam(IC, xr)` plots the free response of a beam at location `x = xr`, where `IC` is a 2-by-*np* matrix specifying the initial disturbances at *np* points along the beam length:

$$\texttt{IC} = \begin{bmatrix} a_0(x_1) & a_0(x_2) & \cdots & a_0(x_{np}) \\ b_0(x_1) & b_0(x_2) & \cdots & b_0(x_{np}) \end{bmatrix}$$

with $a_0(x)$ and $b_0(x)$ being given in Eq. (12.27).

Window 3.3. Functions freebeam and freebeam1 (*Continued*)

`[y, t] = freebeam(IC, xr, tf, npts)` returns the free response of a beam at `xr` in a two-row matrix `y` and times of computation in a vector `t`. Here, `tf` is total computation time and `npts` is the number of time points in the region `[0, tf]`. By default, `tf` is four times the first natural period of the beam and `npts` = 201. With `y` and `t`, the beam free response can be plotted by the MATLAB function `plot` as follows:

```
plot(t, y(1,:))    for displacement
plot(t, y(2,:))    for velocity
```

`freebeam1(IC, ts)` plots the spatial profile of the free displacement of a beam at a specific time `ts`.

`[w, x] = freebeam1(IC, ts, npts)` returns the spatial profile of the free displacement of a beam at time `ts` in a vector `w` and spatial points of computation in a vector `x`, where `npts` is the number of equally-spaced points in the region `[0, L]` for computation. By default, `npts` = 201.

EXAMPLE 3.1

A simply supported beam of parameters $\rho = 0.4$, $EI = 2{,}400$, $L = 7.5$ has viscous damping $d_v = 0.8$. Under the initial disturbances

$$w(x,0) = 0.001x(L-x)^2, \quad \frac{\partial}{\partial t}w(x,0) = 0; \quad 0 \le x \le L$$

the free response of the beam at $x = 3$ is computed in the following three steps:

(i) Set up the system by

```
>> setbeam(0.4, 2400, 7.5, [1 1], 0.8)
```

(ii) Form the initial condition matrix by

```
>> np = 100; xx = 7.5/np*(1:np);
>> A0 = 0.001*xx.*(7.5-xx).^2
>> B0 = zeros(1, np);
>> IC = [A0; B0];
```

The initial displacement can be displayed by `plot(xx, A0)` or `freebeam1(IC, 0)`; see Fig. 12.3.1.

(iii) Compute the free response of the beam at $x = 3$ by

```
>> freebeam(IC, 3)
```

which yields Fig. 12.3.2.

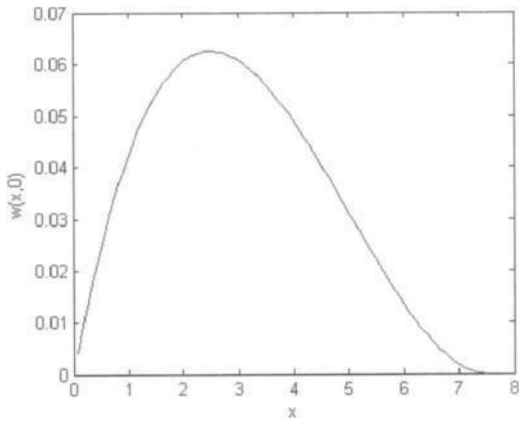


FIGURE 12.3.1 Initial displacement $w(x, 0)$.

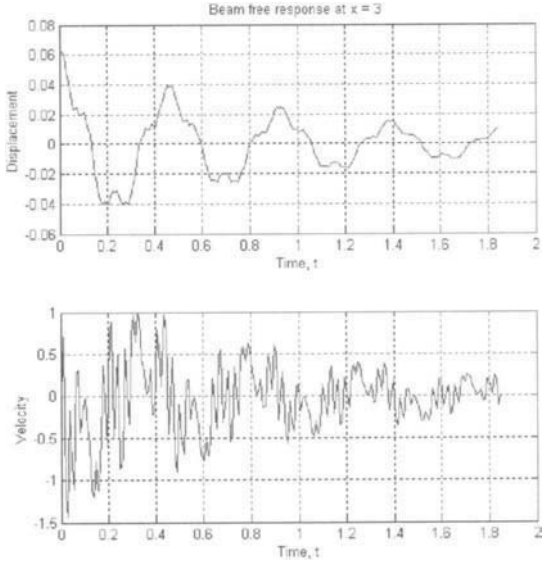


FIGURE 12.3.2 Free response at $x = 3$ with viscous damping.

To set up modal damping, say 5% for each mode, type

```
>> setdamp([1 0.05])
```

This is done without the need to call function `setbeam` again. The free response with the modal damping is plotted in Fig. 12.3.3 by

```
>> freebeam(IC, 3)
```

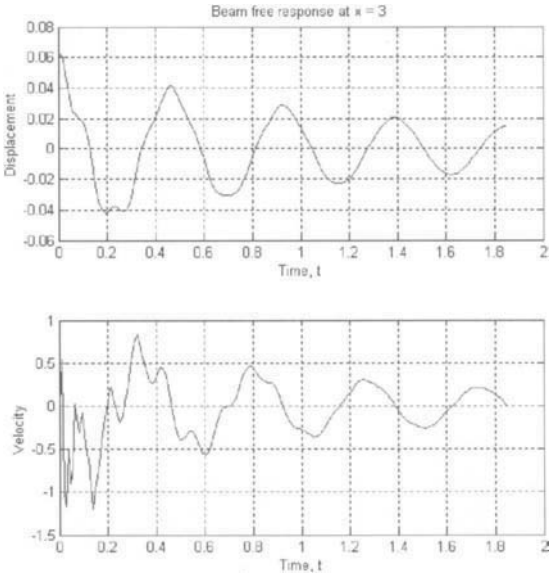


FIGURE 12.3.3 Free response at $x = 3$ with modal damping.

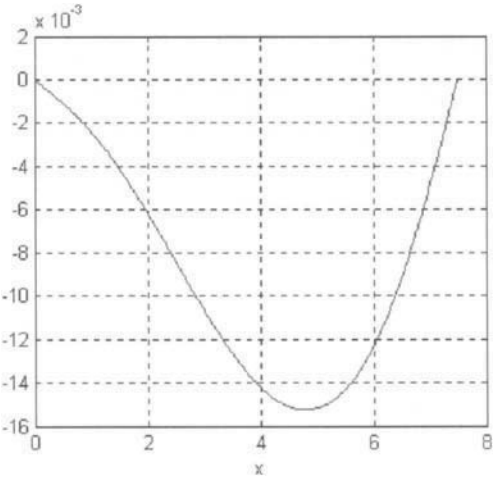


FIGURE 12.3.4 Free displacement profile at $t = 0.14$.

Comparison of Figs. 12.3.2 and 12.3.3 shows the effect of modal damping on higher-frequency modes. In addition, the spatial profile of the free displacement of the beam at $t = 0.14$ is plotted in Fig. 12.3.4 by

```
>> freebeam1(IC, 0.14)
```

12.3.4 Forced Vibration

For a beam under an external load, its modal coordinates are governed by

$$\begin{aligned} \ddot{q}_k(t)+2\xi_k\omega_k\dot{q}_k(t) + \omega_k^2 q_k(t) &= b_k f(t) \\ q_k(0) = 0, \quad \dot{q}_k(0) &= 0 \end{aligned} \tag{12.43}$$

where b_k is related to force distribution as defined in Eq. (12.32). Three types of force distribution are given as follows:

(a) For a pointwise load at $x = x_f$ as shown in Fig. 12.3.5(a), $F(x,t) = \delta(x - x_f) f(t)$ where $\delta(\cdot)$ is the Dirac delta function, and

$$b_k = \int_0^L \delta(x - x_f) u_k(x)\, dx = u_k(x_f) \tag{12.44a}$$

(b) For a force uniformly distributed in the region $x_1 \le x \le x_2$ as shown in Fig. 12.3.5(b), $F(x,t) = (h(x - x_1) - h(x - x_2))\, p(t)$, where $h(x)$ is the unite step function, and

$$b_k = \int_{x_1}^{x_2} u_k(x)\, dx \tag{12.44b}$$

(c) For a pointwise torque at $x = x_\tau$ as shown in Fig. 12.3.5(a), $F(x,t) = -\frac{d\delta(x - x_\tau)}{dx} \tau(t)$, and

$$b_k = -\int_0^L \frac{d\delta(x - x_\tau)}{dx} u_k(x)\, dx = \frac{du_k(x_\tau)}{dx} \tag{12.44c}$$

The solution of Eq. (12.43) is obtained in the following two cases.

Case 1. $\omega_k \neq 0$ (flexible mode):

$$q_k(t) = \frac{b_k}{\omega_{dk}} \int_0^t e^{-\xi_k \omega_k (t-\tau)} \sin \omega_{dk}(t - \tau) f(\tau)\, d\tau; \quad \omega_{dk} = \omega_k \sqrt{1 - \xi_k^2} \tag{12.45}$$

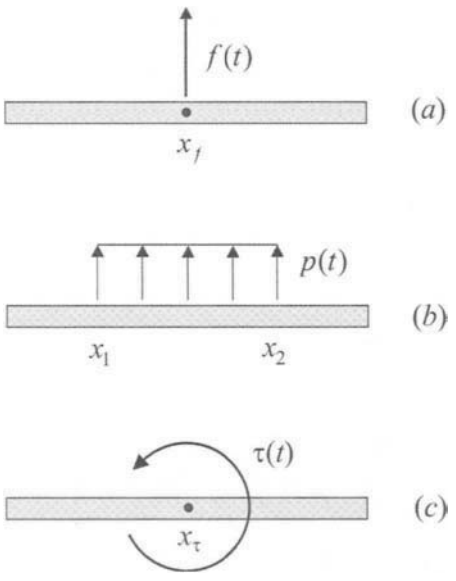


FIGURE 12.3.5 Three types of force distribution.

Case 2. $\omega_k = 0$ (rigid-body mode): If the beam has viscous damping, Eq. (12.39) becomes $\ddot{q}_k(t) + (d_v/\rho)\dot{q}_k(t) = b_k f(t)$, which has the solution

$$q_k(t) = b_k \int_0^t e^{-\sigma(t-\tau)} \int_0^\tau f(\xi) d\xi \, d\tau, \quad \text{with } \sigma = \frac{d_v}{\rho} \tag{12.46}$$

If the beam has modal damping or no damping, Eq. (12.39) reduces to $\ddot{q}_k(t) = b_k f(t)$, which has the solution

$$q_k(t) = b_k \int_0^t \int_0^\tau f(\xi) d\xi \, d\tau = b_k \int_0^t (t-\tau) f(\tau) \, d\tau \tag{12.47}$$

Table 12.3.1 lists the modal coordinates of a beam subject to impulsive, step, and sinusoidal loads, where function $f(t)$ is from Eq. (12.28).

TABLE 12.3.1 Modal Coordinates of a Beam in Certain Cases of Forced Vibration

Load	*Modal Coordinates*
Impulse $f(t) = I_0\delta(t)$	**Flexible mode** ($\omega_k \neq 0$): $q_k(t) = \frac{I_0 b_k}{\omega_{dk}} e^{-\xi_k \omega_k t} \sin \omega_{dk} t$; $\dot{q}_k(t) = -\xi_k \omega_k q_k(t) + I_0 b_k e^{-\xi_k \omega_k t} \cos \omega_{dk} t$
	Rigid-body mode ($\omega_k = 0$): (i) For viscous damping $q_k(t) = I_0 b_k \left(1 - e^{-\sigma t}\right)/\sigma$, $\dot{q}_k(t) = I_0 b_k e^{-\sigma t}$, with $\sigma = d_v/\rho$; (ii) For modal damping or no damping $q_k(t) = I_0 b_k t$, $\dot{q}_k(t) = I_0 b_k$
Step (constant) load $f(t) = F_0$	**Flexible mode** ($\omega_k \neq 0$): $q_k(t) = \frac{F_0 b_k}{\omega_k^2}\left\{1 - e^{-\xi\omega_k t}\left(\cos \omega_{dk} t + \frac{\xi_k \omega_k}{\omega_{dk}} \sin \omega_{dk} t\right)\right\}$; $\dot{q}_k(t) = \frac{F_0 b_k}{\omega_{dk}} e^{-\xi\omega_k t} \sin \omega_{dk} t$
	Rigid-body mode ($\omega_k = 0$): (i) For viscous damping $q_k(t) = \frac{F_0 b_k}{\sigma}\left(t - \frac{1}{\sigma}\left(1 - e^{-\sigma t}\right)\right)$, $\dot{q}_k(t) = \frac{F_0 b_k}{\sigma}\left(1 - e^{-\sigma t}\right)$, with $\sigma = \frac{d_v}{\rho}$; (ii) For modal damping or no damping $q_k(t) = \frac{1}{2} F_0 b_k t^2$, $\dot{q}_k(t) = F_0 b_k t$

TABLE 12.3.1 Modal Coordinates of a Beam in Certain Cases of Forced Vibration (*Continued*)

Load	*Modal Coordinates*
Sinusoidal load $f(t) = A\sin(\omega t + \phi_0)$	**Flexible mode** ($\omega_k \neq 0$):

(i) Nonresonant vibration ($\xi_k \neq 0$ or $\omega \neq \omega_k$)

$$q_k(t) = Ab_kH_k \sin(\omega t + \phi_0 - \psi_k) - Ab_kH_k e^{-\xi_k\omega_k t}(\sin(\phi_0 - \psi_k)\cos\omega_{dk}t + \gamma_k \sin\omega_{dk}t)$$

$$\dot{q}_k(t) = Ab_kH_k\omega\cos(\omega t + \phi_0 - \psi_k) - Ab_kH_k e^{-\xi_k\omega_k t}\omega\cos(\phi_0 - \psi_k)\cos\omega_{dk}t + Ab_kH_k e^{-\xi_k\omega_k t}(\xi_k\omega_k\gamma_k + \omega_{dk}\sin(\phi_0 - \psi_k))\sin\omega_{dk}t$$

where $$H_k = \frac{1}{\sqrt{(\omega_k^2 - \omega^2)^2 + 4\xi_k^2\omega_k^2\omega^2}}, \quad \psi_k = \tan^{-1}\left(\frac{2\xi_k\omega_k}{\omega_k^2 - \omega^2}\right)$$
$$\gamma_k = \{\omega\cos(\phi_0 - \psi_k) + \xi_k\omega_k\sin(\phi_0 - \psi_k)\}/\omega_{dk}$$

(ii) Resonant vibration ($\xi_k = 0$ and $\omega = \omega_k$)

$$q_k(t) = \frac{Ab_k}{2\omega_k^2}\{-\omega_k t\cos(\omega_k t + \phi_0) + \cos(\phi_0)\sin\omega_k t\}$$

$$\dot{q}_k(t) = \frac{Ab_k}{2\omega_k}\{\sin(\phi_0)\sin\omega_k t + \omega_k t\sin(\omega_k t + \phi_0)\}$$

Rigid-body mode ($\omega_k = 0$):

(i) For viscous damping

$$q_k(t) = \frac{Ab_k}{\sigma\omega(\sigma^2 + \omega^2)}\left\{\omega e^{-\sigma t}(\sigma\sin\phi_0 - \omega\cos\phi_0) + (\sigma^2 + \omega^2)\cos\phi_0 - \omega\sigma\sin(\omega t + \phi_0) - \sigma^2\cos(\omega t + \phi_0)\right\}$$

$$\dot{q}_k(t) = \frac{Ab_k}{\sqrt{\sigma^2 + \omega^2}}\left\{\sin(\omega t + \phi_0 - \theta_0) - e^{-\sigma t}\sin(\phi_0 - \theta_0)\right\}$$

with $\sigma = d_v/\rho$, $\theta_0 = \tan^{-1}(\omega/\sigma)$

(ii) For modal damping or no damping

$$q_k(t) = \frac{Ab_k}{\omega^2}\{\sin\phi_0 - \sin(\omega t + \phi_0) + \omega t\cdot\cos\phi_0\}$$

$$\dot{q}_k(t) = \frac{Ab_k}{\omega}\{\cos\phi_0 - \cos(\omega t + \phi_0)\}$$

The Toolbox of this chapter provides functions `forcedbeam` and `forcedbeam1` for computing forced vibration of a beam; see Window 3.4.

Window 3.4. Functions `forcedbeam` and `forcedbeam1`

MATLAB Functions: **`forcedbeam, forcedbeam1`**

Purpose:
To compute and display forced response of a beam.

Window 3.4. Functions forcedbeam and forcedbeam1 (*Continued*)

Synopsis:

```
forcedbeam(Load_Spec, xr)
[y, t] = forcedbeam(Load_Spec, xr, tf, npts)
forcedbeam1(Load_Spec, t)
[w, x] = forcedbeam1(Load_Spec, t, npts)
```

Description:

`forcedbeam(Load_Spec, xr)` plots the forced response of a beam at `x = xr`, where `Load_Spec` is a vector specifying one external load according to Table 12.3.2.

`[y, t] = forcedbeam(Load_Spec, xr, tf, npts)` returns the forced response of a beam at `xr` in a two-row matrix `y` and times of computation in a vector `t`. Here, `tf` is total computation time and `npts` is the number of time points in the region `[0, tf]`. By default, `tf` is four times the first natural period of the beam and `npts` = 201. With `y` and `t`, the forced response of the beam can be plotted by the MATLAB function `plot` as follows:

`plot(t, y(1,:))`	for displacement
`plot(t, y(2,:))`	for velocity

`forcedbeam1(Load_Spec, t)` plots the spatial distribution of the forced response of a beam at a specific time `t`. Refer to Table 12.3.2 for assignment of `Load_Spec`.

`[w, x] = forcedbeam1(Load_Spec, t, npts)` returns the spatial distribution of the forced response in a vector `w` and spatial points of computation in a vector `x`, where `npts` is the number of equally-spaced points in the region `[0, L]` for computation. By default, `npts` = 201.

TABLE 12.3.2 External Loads and Load Specification Vector `Load_Spec`

External Load	*F(x,t)*	`Load_Spec`
(L0) Pointwise impulse force	$I_0\delta(t)\delta(x-x_f)$	[0 x_f I_0]
(L1) Pointwise step force	$F_0\delta(x-x_f)$	[1 x_f F_0]
(L2) Pointwise sinusoidal force	$A\sin(\omega t+\phi_0)\delta(x-x_f)$	[2 x_f A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
(L10) Uniformly distributed impulse force	$I_0\delta(t)$, for $x_1 \le x \le x_1$	[10 x_1 x_2 I_0]
(L11) Uniformly distributed step force	F_0, for $x_1 \le x \le x_1$	[11 x_1 x_2 F_0]
(L12) Uniformly distributed sinusoidal force	$A\sin(\omega t+\phi_0)$, for $x_1 \le x \le x_1$	[12 x_1 x_2 A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
(L20) Pointwise impulse torque	$-I_0\delta(t)\dfrac{d}{dx}\delta(x-x_\tau)$	[20 x_τ I_0]
(L21) Pointwise step torque	$-\tau_0\dfrac{d}{dx}\delta(x-x_\tau)$	[21 x_τ τ_0]
(L22) Pointwise sinusoidal torque	$-A\sin(\omega t+\phi_0)\dfrac{d}{dx}\delta(x-x_\tau)$	[22 x_τ A ω ϕ_0] ω in rad/s, ϕ_0 in degrees

EXAMPLE 3.2

In Fig. 12.3.6, a beam ($\rho = 0.4$, $EI = 1{,}500$, $L = 4$) with clamped and sliding-elastic ends ($k_t = 10$) is subject to a pointwise external load $f(t)$ at $x_f = 1.7$. The beam has viscous damping with $d_v = 1.25$. If the external load is an impulse, $f(t) = -2\delta(t)$, the dynamic response of the beam at $x = 3.5$ is obtained by

```
>> setbeam(0.4, 1500, 4, [2 7 10], 1.25)
>> forcedbeam([0 1.7 -2], 3.5, 3)
```

as shown Fig. 12.3.7.

Function `forcedbeam1` can be used to examine the beam displacement profile immediately after the impulse is applied. Consider the beam displacement distribution at

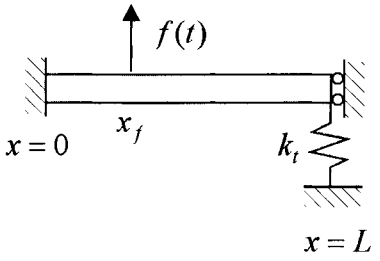


FIGURE 12.3.6

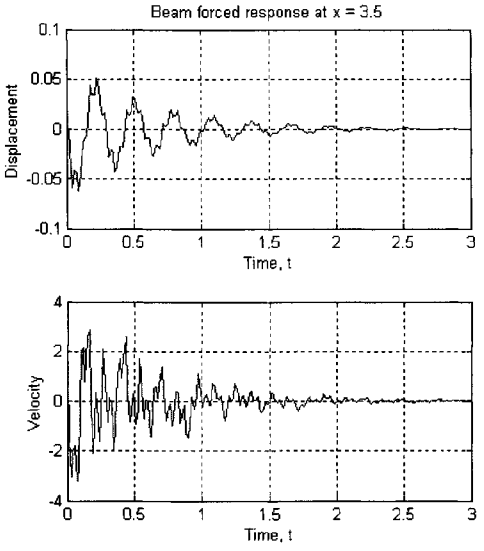


FIGURE 12.3.7

times $t = j \times \Delta t, j = 0, 1, \ldots, 20$, where $\Delta t = T_1/1{,}280$ with $T_1 = 2\pi/\omega_1 = 0.288589$ being the first natural period of the beam. By the commands typed in an M-file:

```
% m-file for computing beam profile
P = 0.288589/1280;
Load = [0 1.7 -2];
[y, x] = forcedbeam1(Load,0);
clf
plot(x,y (1,:)); grid
hold
for i = 1:20
   y = forcedbeam1(Load, i*P);
   plot(x, y(1,:))
end
xlabel('x')
title('Distribution of Displacement')
hold
```

the beam profiles at those times are plotted in Fig. 12.3.8.

Now let the external load be sinusoidal, $f(t) = 5\sin(\omega t)$ with $\omega = 21.77$. The excitation frequency is very close to the first natural frequency of the beam, $\omega_1 = 21.7721$. (The beam natural frequencies are given by function `setbeam`.) The forced response of the beam at $x = 4$ is obtained by

```
>> ld = [2 1 5 21.77 0];
>> [y, t] = forcedbeam(ld, 4, 4);
>> clf
>> plot(t, y(1,:)); grid
>> xlabel('x'); title('Beam displacement at x = 4')
```

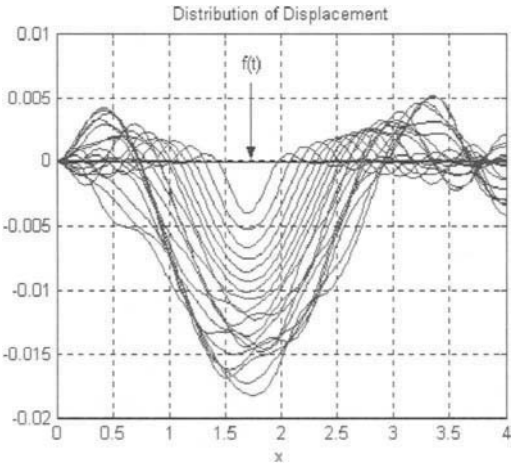


FIGURE 12.3.8

and is shown in Fig. 12.3.9. Furthermore, if the beam is undamped ($d_v = 0$),

```
>> setdamp(0);
>> [y,t]=forcedbeam(ld, 4, 4);
>> plot(t,y(1,:)); grid
>> xlabel('x'); title('Beam displacement at x = 4')
```

produces the resonant response of the beam as shown in Fig. 12.3.10.

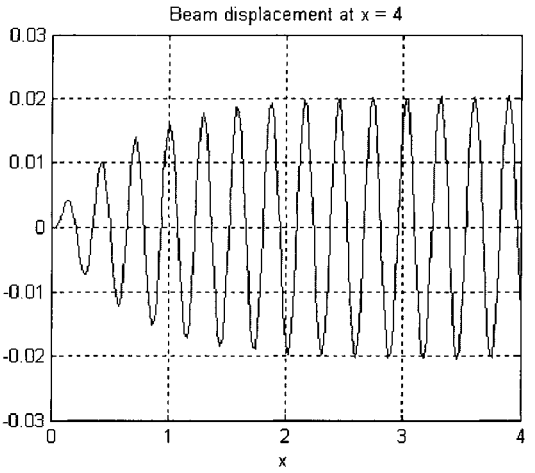


FIGURE 12.3.9

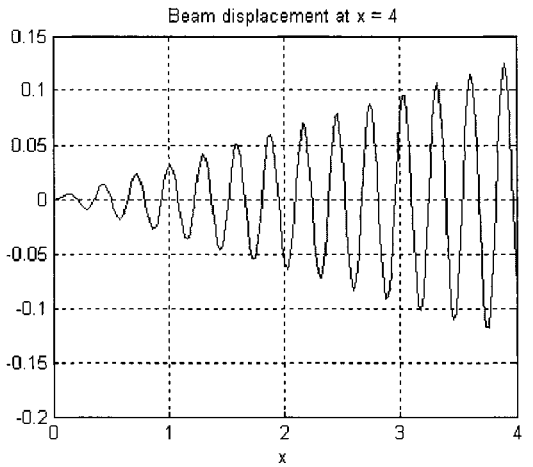


FIGURE 12.3.10

12.3.5 Total Response

According to Sections 12.3.3 and 12.3.4, the total response of a beam subject to both initial and external disturbances is given by Eq. (12.35) with the modal coordinates

$$q_k(t) = e^{-\xi_k \omega_k t}\left(q_k(0)\cos\omega_{dk}t + \frac{\dot{q}_k(0) + \xi_k\omega_k q_k(0)}{\omega_{dk}}\sin\omega_{dk}t\right) + \frac{b_k}{\omega_{dk}}\int_0^t e^{-\xi_k\omega_k(t-\tau)}\sin\omega_{dk}(t-\tau)f(\tau)\,d\tau \tag{12.48a}$$

for flexible modes ($\omega_k \neq 0$), and

$$q_k(t) = q_k(0) + \dot{q}_k(0)\,t + b_k\int_0^t\int_0^\tau f(\xi)\,d\xi\,d\tau \tag{12.48b}$$

for rigid-body modes ($\omega_k = 0$).

Determination of the total response by the Toolbox is presented in Window 3.5. Besides the above modal series solution, the dynamic response of a beam can also be determined by numerical methods. This Toolbox also has a function `contrbeam` for computing beam dynamic response by a Runge-Kutta numerical integration algorithm; see Window 4.4.

Window 3.5. Determination of Total Response of a Beam

Disturbances:

Initial disturbances $w(x,0) = a_0(x)$ and $\frac{\partial}{\partial t}w(x,0) = b_0(x)$, described by matrix `IC`

External load $F(x,t) = D(x)f(t)$, described by vector `Load_Spec`

Method 1: To determine time response of a beam at location `xr`

```
y = freebeam(IC, xr) + forcedbeam(Load_Spec, xr);
plotbeam(y, opt)
```

Method 2: To determine spatial distribution of beam displacement at a time `ts`

```
w = freebeam1(IC, ts) + forcedbeam1(Load_Spec, ts);
plotbeam1(w)
```

Here `plotbeam` and `plotbeam1` are two functions given in Window 3.6.

Window 3.6. Functions `plotbeam` and `plotbeam1`

MATLAB Functions: **`plotbeam, plotbeam1`**

Purpose:

To plot the response of a beam computed by `freebeam` and `forcedbeam` or by `freebeam1` and `forcedbeam1`.

Window 3.6. Functions `plotbeam` and `plotbeam1` (*Continued*)

Synopsis:

```
plotbeam(y)
plotbeam(y, opt)
plotbeam1(w)
```

Description:

`plotbeam(y)` plots the displacement of a beam versus time, where `y` is a two-row matrix of beam response obtained by `freebeam` and/or `forcedbeam` (see Windows 3.3 and 3.4).

`plotbeam(y, opt)` plots the response (displacement and velocity) of a beam versus time, with the following options:

`opt = 0` plot displacement only

`opt = 1` plot velocity only

`opt = 2` plot both displacement and velocity

`plotbeam1(w)` plots the displacement of a beam versus spatial coordinate x, where `w` is a vector of beam displacement obtained by `freebeam1` and/or `forcedbeam1` (see Windows 3.3 and 3.4).

EXAMPLE 3.3

A beam ($\rho = 1, EI = 100, L = 3$) with pinned-elastic ($k_r = 50$) and pinned ends is subject to the initial conditions

$$w(x,0) = 0.01x, \quad \frac{\partial}{\partial t} w(x,0) = 0$$

and a distributed step load

$$F(x,t) = \begin{cases} 0 & \text{for} \quad 0 \le x < 1 \\ -0.1 & \text{for} \quad 1 \le x \le 3 \end{cases}$$

Assume that there exists 5% modal damping in every mode of the beam. Use the first 20 modes for simulation. The total beam response (displacement and velocity) at $x = 3$ is plotted in Fig. 12.3.11 by

```
>> setbeam(1, 100, 3, [6 50 3], [1 0.05], 20)
>> xx = 3*[1:1000]/1000;
>> A0 = 0.01*xx; B0 = 0*xx; IC = [A0; B0];
>> Load = [11 1 3 -0.1];
>> y = freebeam(IC, 2, 20, 2001) + forcedbeam(Load, 2, 20, 2001);
>> plotbeam(y, 2)
```

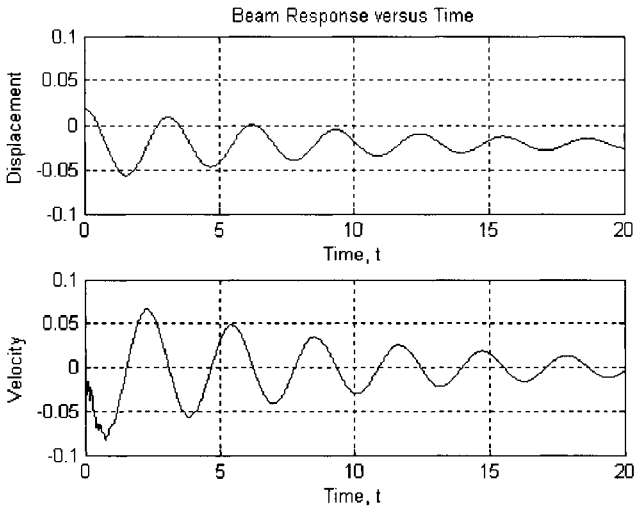


FIGURE 12.3.11

12.3.6 Animation of Time Response

The Toolbox of this chapter provides function `beammovie` for animating the time response of a beam under initial and external disturbances; see Window 3.7.

Window 3.7. Function beammovie

MATLAB Function: **beammovie**

Purpose:
To animate time response of a beam in its spatial region $0 \leq x \leq L$.

Synopsis:

```
beammovie(Load_Spec)
beammovie(Load_Spec, IC)
beammovie(Load_Spec, IC, IS_control, n_frames, nfpp, frame_bounds)
F = beammovie(Load_Spec, IC, IS_control, n_frames, nfpp,
frame_bounds)
```

Description:
`beammovie(Load_Spec)` plays the animated vibration of a beam subject to an external load that is specified by vector `Load_Spec` according to Table 12.3.2.

`beammovie(Load_Spec, IC)` plays the animated vibration of a beam subject to an external load specified by vector `Load_Spec`, and initial disturbances that are specified by matrix `IC` according to Window 3.3. If there is no external load, set `Load_Spec = 0` or `[ ]`.

Window 3.7. Function beammovie (*Continued*)

`beammovie(Load_Spec, IC, IS_control, n_frames, nfpp, frame_bounds)` plays the animated vibration of a beam in a specified manner, where `IS_control` is an animation control parameter with the options

`IS_control = 0` play frames continuously
`IS_control = 1` play frames one by one

`n_frames` is the total number of frames in the animation, `nfpp` is the number of frames per first damped period of the beam, and vector `frame_bounds = [xmin xmax ymin ymax]` sets up the bounds of frames. By default, `IS_control = 0`, `n_frames` = 97, `nfpp` = 12, and `frame_bounds` is set up according to the maximum and minimum of beam vibration within the first natural or damped period. The `IS_control`, `n_frames`, `nfpp`, and `frame_bounds` can be replaced by [ ] as a placeholder for their default values. If there is no external load, `Load_Spec = 0` or [ ]. On the other hand, if there are no initial disturbances, `IC = 0` or [ ].

`F = beammovie(Load_Spec, IS_control, n_frames, nfpp, frame_bounds)` returns a set of frames of animated beam vibration in a matrix `F`, which can be played back by the MATLAB function `movie(F)`.

EXAMPLE 3.4

A simply-supported beam of parameters $\rho = 0.4$, $EI = 2{,}400$, and $L = 6$ is subject to a pointwise constant load $f(t) = -10$ at $x = 4.5$, and initial disturbances $w(x, 0) = 0.001x(L - x)^2$, $\frac{\partial}{\partial t}w(x, 0) = 0$. Assume that the beam has viscous damping with $d_v = 0.8$. The first damped period of the beam, by function `setbeam`, is $T_1 = 0.2962$. The forced vibration of the beam is animated by the commands

```
>> setbeam(0.4, 2400,6,[1 1], 0.8)
>> xx = 6/1000*(1:1000); A0 = 0.001*xx .*(6-xx).^2; B0 = 0*A0;
>> IC = [A0; B0]; % initial conditions
>> LD_Spec = [1 4.5 -10]; % external load
>> beammovie(LD_Spec, IC, 1, 9, 8, [0 6 -0.05 0.05])
```

which produce 9 frames of the beam displacement profiles. Shown in Fig. 12.3.12 are the first 8 frames at times $t_i = i \times h$, $i = 0, 1, 2, \ldots, 7$, with $h = T_1/8 = 0.037025$.

12.3.7 Frequency Response

Consider a beam subject to a pointwise sinusoidal force at $x = x_f$:

$$\rho \frac{\partial^2}{\partial t^2} w(x,t) + d_v \frac{\partial}{\partial t} w(x,t) + EI \frac{\partial^4}{\partial x^4} w(x,t) = f_0 e^{j\omega t} \delta(x - x_f), \quad 0 \le x \le L \tag{12.49}$$

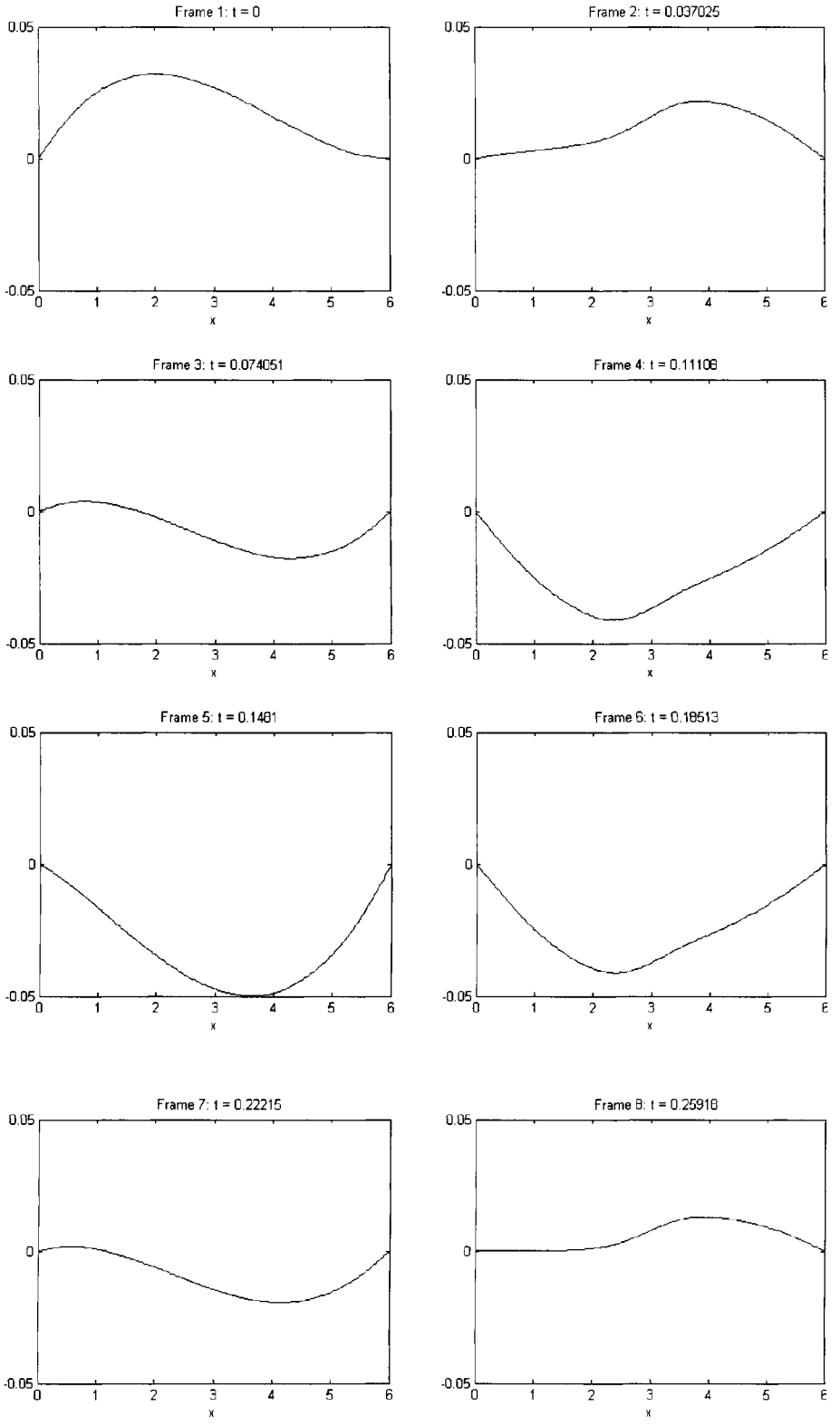


FIGURE 12.3.12 Frames of the animated beam displacement in Example 3.4.

where $j = \sqrt{-1}$ and ω is the excitation frequency. Besides viscous damping, modal damping can also be specified; see Window 3.1. The steady-state solution of Eq. (12.49) is expressed by an N-term modal series

$$w_{ss}(x,t) = \sum_{k=1}^{N} u_k(x) q_k(t) \tag{12.50}$$

where the modal coordinates are governed by the differential equations

$$\ddot{q}_k(t) + 2\xi_k \omega_k \dot{q}_k(t) + \omega_k^2 q_k(t) = b_k f_0 \, e^{j\omega t}, \quad k = 1, 2, \ldots \tag{12.51}$$

with $b_k = u_k(x_f)$. For steady-state vibration, $q_k(t)$ is assumed as

$$q_k(t) = Q_k \, e^{j\omega t} \tag{12.52}$$

which, by Eq. (12.51) leads to

$$Q_k = \frac{f_0 b_k}{\omega_k^2 - \omega^2 + j2\xi_k \omega_k \omega} \tag{12.53}$$

It follows that the steady-state displacement of the beam is

$$w_{ss}(x,t) = W(x,\omega) \, e^{j\omega t} \tag{12.54}$$

where

$$W(x,\omega) = f_0 \sum_{k=1}^{N} \frac{b_k}{\omega_k^2 - \omega^2 + j2\xi_k \omega_k \omega} u_k(x) \tag{12.55}$$

The $W(x,\omega)$ is called the *frequency response* of the beam. Rewrite Eq. (12.54) as

$$w_{ss}(x,t) = H \, e^{j(\omega t + \phi)} \tag{12.56}$$

where H and ϕ are the magnitude and phase of the frequency response

$$H = |W(x,\omega)|, \quad \phi = \angle W(x,\omega) \tag{12.57}$$

The above result is obtained based on a sinusoidal force of complex format ($e^{j\omega t}$). The frequency response $W(x,\omega)$ is applicable to sinusoidal forces of real format. For instance, the steady-state response of a beam under $F(x,t) = f_0 \sin(\omega t + \phi_0)\delta(x - x_f)$ is

$$w_{ss}(x,t) = H \sin(\omega t + \phi_0 + \phi) \tag{12.58}$$

The Toolbox of this chapter provides a function `frfbeam` for plotting the frequency response of a beam subject to a pointwise sinusoidal load; see Window 3.8.

Window 3.8. Function `frfbeam`

MATLAB Function: **`frfbeam`**

Purpose:
To plot the magnitude and phase of steady-state displacement of a beam subject to a pointwise sinusoidal load.

Synopsis:

```
frfbeam(xr, xf, f0)
frfbeam(xr, xf)
[y, freq] = frfbeam(xr, xf, f0, omg0, omgf, npts)
```

Description:
`frfbeam(xr, xf, f0)` plots the magnitude and phase of steady-state displacement of a beam at `xr` versus excitation frequency. The beam is under a sinusoidal load of amplitude `f0`, applied at `xf`. The magnitude and phase of the beam frequency response are defined in Eq. (12.57).

`frfbeam(xr, xf)` plots the beam frequency response for `f0` = 1.

`[y, freq] = frfbeam(xr, xf, f0, omg0, omgf, npts)` returns the frequency response in a two-row matrix `y` and frequencies of simulation in a vector `freq`. Here, `omg0` and `omgf` are the lower and upper bounds a of frequency region, and `npts` is the number of frequency points in the region `[omg0, omgf]` for simulation. By default, `omg0` = 0, `omgf` is larger than the third or fourth natural frequency of the beam and `npts` = 1,001. With `y` and `freq`, the frequency response can be plotted by the MATLAB function `plot` as follows:

for magnitude	`plot(freq, y(1,: ))`
for phase	`plot(freq, y(2,: ))`

EXAMPLE 3.5

A pinned-clamped beam of parameters $\rho = 10$, $EI = 5{,}000$, $L = 6$ is subject to a sinusoidal force of amplitude $f_0 = 15$ at $x_f = 4.2$. Assume that each mode of the beam has 1% modal damping. By

```
>> setbeam(10, 5000, 6, [1 2], [1 0.01])
>> frfbeam(1.8, 4.2, 15)
```

the frequency response of the beam at point $x = 1.8$ is plotted in Fig. 12.3.13.

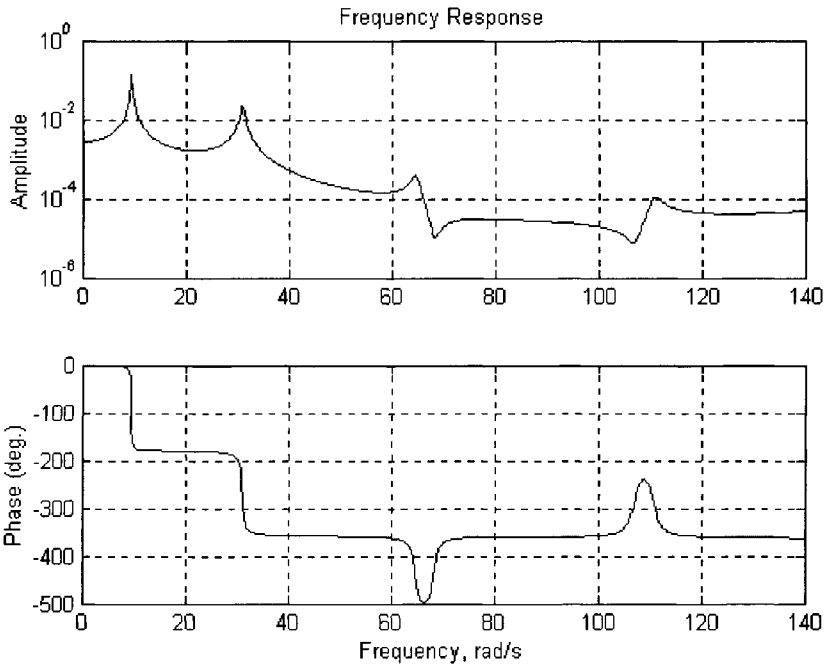


FIGURE 12.3.13

12.4 Feedback Control

In this section, control system formulation for uniform beams with pointwise sensors and actuators is considered; the related MATLAB functions from the Toolbox are introduced.

12.4.1 Control System Formulation

Governing Equations In Fig. 12.4.1, a uniform beam is under feedback control, where the control system has r pointwise sensors located at $s_1, s_2, \ldots, s_r$, and s pointwise actuators

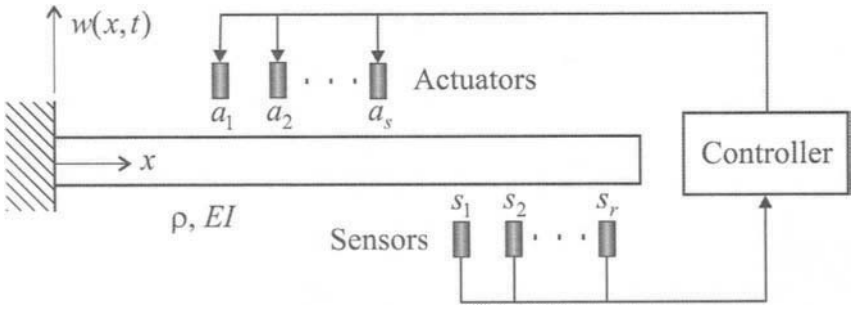


FIGURE 12.4.1 Schematic of control system for a beam.

TABLE 12.4.1 Actuator and Control Force

Actuator Type	*Vector* $\{\alpha_j\}$	`Actuator_j`
(A1) Transverse force $F_{c,j}(x,t) = f_j(t)\delta(x-a_j)$	$\{\alpha_j\} = [\Phi(a_j)]$	[1 a_j]
(A2) Moment $F_{c,j}(x,t) = -f_j(t)\dfrac{d}{dx}\delta(x-a_j)$	$\{\alpha_j\} = \dfrac{d}{dx}[\Phi(a_j)]$	[2 a_j]

located at $a_1, a_2, \ldots, a_s$. The transverse displacement of the beam is governed by

$$\rho\frac{\partial^2}{\partial t^2}w(x,t) + d_v\frac{\partial}{\partial t}w(x,t) + EI\frac{\partial^4}{\partial x^4}w(x,t) = F_{ext}(x,t) + \sum_{j=1}^{s} F_{c,j}(x,t), \quad 0 \le x \le L \tag{12.59}$$

where $F_{ext}(x,t)$ is an external force (not shown in Fig. 12.4.1), $F_{c,j}(x,t)$ is the control force applied to the beam by the jth actuator. It is assumed that external force is of the form

$$F_{ext}(x,t) = D(x)f_e(t) \tag{12.60}$$

where $D(x)$ describes the spatial distribution of the force.

Listed in Table 12.4.1 are two types of actuators:

(A1) The jth actuator applies a transverse load to the beam

$$F_{c,j}(x,t) = f_j(t)\delta(x-a_j) \tag{12.61a}$$

(A2) The jth actuator applies a moment to the beam,

$$F_{c,j}(x,t) = -f_j(t)\frac{d}{dx}\delta(x-a_j) \tag{12.61b}$$

The sensor output signals can be written as

$$\{y_s(t)\} = (y_1(t) \quad y_2(t) \quad \cdots \quad y_r(t))^T \tag{12.62}$$

where $y_j(t)$ is the output of the jth sensor. Listed in Table 12.4.2 are six types of sensors (S1 to S6) and associate sensor outputs. For example, if the jth sensor measures the transverse displacement of the beam, $y_j(t) = w(s_j, t)$.

Modal Expansion Consider the first n modes of the beam in control system formulation. Following Section 12.3.1, the beam displacement is approximated by

$$w(x,t) \approx [\Phi(x)]^T\{q(t)\} \tag{12.63}$$

with

$$\begin{aligned}[\Phi(x)] &= (u_1(x) \quad u_2(x) \quad \cdots \quad u_n(x))^T \\ \{q(t)\} &= (q_1(t) \quad q_2(t) \quad \cdots \quad q_n(t))^T\end{aligned} \tag{12.64}$$

TABLE 12.4.2 Sensor and Output Signal

Sensor Type	*Vectors $\{\beta_j\}$ and $\{\gamma_j\}$*	`Sensor_j`
(S1) Transverse displacement $y_j(t) = w(s_j, t)$	$\{\beta_j\} = [\Phi(s_j)]$, $\{\gamma_j\} = \{0\}$	[1 s_j]
(S2) Slope or rotation $y_j(t) = \theta(s_j, t) = \frac{\partial}{\partial x} w(s_j, t)$	$\{\beta_j\} = \frac{d}{dx}[\Phi(s_j)]$, $\{\gamma_j\} = \{0\}$	[2 s_j]
(S3) Strain $y_j(t) = \frac{\partial^2}{\partial x^2} w(s_j, t) = \kappa(s_j, t)$	$\{\beta_j\} = \frac{d^2}{dx^2}[\Phi(s_j)]$, $\{\gamma_j\} = \{0\}$	[3 s_j]
(S4) Transverse velocity $y_j(t) = \frac{\partial}{\partial t} w(s_j, t)$	$\{\beta_j\} = \{0\}$, $\{\gamma_j\} = [\Phi(s_j)]$	[4 s_j]
(S5) Rotation velocity $y_j(t) = \frac{\partial}{\partial t} \theta(s_j, t)$	$\{\beta_j\} = \{0\}$, $\{\gamma_j\} = \frac{d}{dx}[\Phi(s_j)]$	[5 s_j]
(S6) Rate of change of strain $y_j(t) = \frac{\partial}{\partial t} \kappa(s_j, t)$	$\{\beta_j\} = \{0\}$, $\{\gamma_j\} = \frac{d^2}{dx^2}[\Phi(s_j)]$	[6 s_j]

where $u_k(x)$ are the eigenfunctions of the beam defined by Eqs. (12.6) and (12.7), and $q_k(t)$ are unknown modal coordinates to be determined. Substitution of Eq. (12.63) into Eq. (12.59) and use of the orthonormal conditions (12.9) yields a discretized model of the controlled beam:

$$\{\ddot{q}(t)\} + [D_b]\{\dot{q}(t)\} + [\Omega_b]\{q(t)\} = \{B_f\} f_e(t) + [B_c]\{U(t)\} \tag{12.65}$$

where $\{U(t)\}$ is a control force vector of the form

$$\{U(t)\} = (f_1(t) \quad f_2(t) \quad \cdots \quad f_s(t))^T \tag{12.66}$$

$[D_b]$ and $[\Omega_b]$ are damping and stiffness matrices given by

$$[D_b] = \begin{cases} \dfrac{d_v}{\rho}[I] & \text{for the beam with viscous damping} \\ \underset{i}{\text{diag}}\{2\xi_i\omega_i\} & \text{for the beam with modal damping} \end{cases} \tag{12.67}$$

$$[\Omega_b] = \underset{k}{\text{diag}}\{\omega_k^2\}$$

with ω_k being the natural frequencies of the beam defined in Eq. (12.6) and $[I]$ an identity matrix, and $\{B_f\}$ is given by

$$\{B_f\} = (b_1 \quad b_2 \quad \cdots \quad b_n)^T, \quad \text{with } b_k = \int_0^L D(x) u_k(x)\, dx \tag{12.68}$$

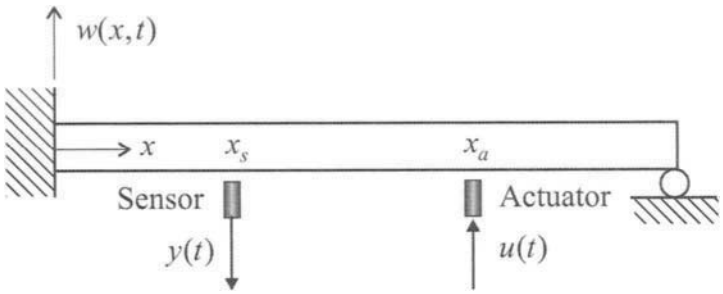


FIGURE 12.4.2 A beam controlled via a pair of sensor and actuator.

For a pointwise external force at x_f, $\{B_f\} = [\Phi(x_f)]$. Matrix $[B_c]$ is composed of

$$[B_c] = [\{\alpha_1\} \quad \{\alpha_2\} \quad \cdots \quad \{\alpha_s\}] \in R^{n\times s} \tag{12.69}$$

where the jth column vector $\{\alpha_j\}$ is dependent upon the location and type of the jth actuator. The form of $\{\alpha_j\}$ for force-type (A1) and moment-type (A2) actuators is given in Table 12.4.1.

Open-Loop with a Pair of Sensor and Actuator As a special case, consider the beam controlled via a sensor located at $x = x_s$ and an actuator located at $x = x_a$; see Fig. 12.4.2. The transverse displacement of the beam is governed by

$$\rho\frac{\partial^2}{\partial t^2}w(x,t) + d_v\frac{\partial}{\partial t}w(x,t) + EI\frac{\partial^4}{\partial x^4}w(x,t) = (-1)^{\mu}\frac{d^{\mu}\delta(x-x_a)}{dx^{\mu}}u(t), \quad 0 \le x \le L \tag{12.70}$$

where $u(t)$ is the control force or moment by the actuator, with $\mu = 0$ for a force (A1 in Table 12.4.1) and $\mu = 1$ for a moment (A2 in Table 12.4.1). The system output $y(t)$ depends on the type of the senor, and is in one of the following three forms.

(i) For displacement measurement (S1 in Table 12.4.2):

$$y(t) = w(x_s,t) \tag{12.71a}$$

(ii) For slope or rotation measurement (S2 in Table 12.4.2):

$$y(t) = \theta(x_s,t) = \frac{\partial}{\partial x}w(x_s,t) \tag{12.71b}$$

(iii) For strain measurement (S3 in Table 12.4.2):

$$y(t) = \frac{\partial^2}{\partial x^2}w(x_s,t) \tag{12.71c}$$

By Eqs. (12.63) and (12.70), the open-loop transfer of the control system from u to y is obtained as

$$G_o(s) = \frac{Y(s)}{U(s)} = \sum_{k=1}^{n}\frac{a_k b_k}{s^2 + c_k s + \omega_k^2} \tag{12.72}$$

where

$$a_k = \begin{cases} u_k(x_s) & \text{for a displacement sensor (S1)} \\ u_k'(x_s) & \text{for a slope sensor (S2)} \\ u_k''(x_s) & \text{for a strain sensor (S3)} \end{cases}$$

$$b_k = \begin{cases} u_k(x_a) & \text{for a force actuator (A1)} \\ u_k'(x_a) & \text{for a moment actuator (A2)} \end{cases} \tag{12.73}$$

$$c_k = \begin{cases} \dfrac{d_v}{\rho} & \text{for the beam with viscous damping} \\ 2\xi_k\omega_k & \text{for the beam with modal damping} \end{cases}$$

with $u' = du/dx$.

If the coefficient a_k in Eq. (12.72) is zero, the sensor is located at a nodal point of the kth mode shape $u_k(x)$ or its derivatives, and cannot pick up the information on the vibration of the beam in that mode. Likewise, if the coefficient b_k is zero, the actuator is located at a nodal point of $u_k(x)$ or its first derivative, and cannot affect the kth mode of vibration of the beam. Thus, for observability and controllability, locate the sensor and actuator away from the nodal points of those modes of interest.

The sensor and actuator are called *colocated* if $a_k = b_k$ for all k or if $a_k b_k$ has the same sign for all k. It can be shown that the sensor and actuator are colocated if they are at the same location ($x_s = x_a$) and if they are of the same type (i.e., S1 and A1 or S2 and A2). A control system with colocated sensor and actuator are called *colocated control system*. For an undamped beam with colocated sensor and actuator, the open-loop transfer function becomes

$$G_o(s) = \sum_{k=1}^{n} \frac{a_k^2}{s^2 + \omega_k^2} \tag{12.74}$$

This transfer function has $2n$ imaginary poles $\pm j\omega_k$, $k = 1, 2, \ldots, n$, $j = \sqrt{-1}$, and up to $2n - 2$ imaginary zeros $\pm j\alpha_k$, $k = 1, 2, \ldots, n-1$. If the sensor/actuator location is not any nodal point of the modes considered (i.e., $a_k \neq 0$ for all k), the poles and zeros are alternatively located on the imaginary axis; that is,

$$\omega_1 < \alpha_1 < \omega_2 < \cdots < \alpha_{n-1} < \omega_n \tag{12.75}$$

This interesting phenomenon is known as pole-zero alternation. (Also see Section 11.5.2.)

State Equation Equation (12.65) is cast into

$$\{\dot{z}(t)\} = [A]\{z(t)\} + [B]\{U(t)\} + \{B_e\} f_e(t) \tag{12.76}$$

where the state vector

$$\{z(t)\} = \begin{pmatrix} \{q(t)\} \\ \{\dot{q}(t)\} \end{pmatrix} \tag{12.77}$$

and

$$[A] = \begin{bmatrix} [0] & [I] \\ -[\Omega_b] & -[D_b] \end{bmatrix}, \quad [B] = \begin{bmatrix} [0] \\ [B_c] \end{bmatrix}, \quad \{B_e\} = \begin{pmatrix} \{0\} \\ \{B_f\} \end{pmatrix} \tag{12.78}$$

with [0] and {0} being zero matrix and vector of appropriate order. The sensor output equation (12.62) is also written in a state form

$$\{y_s(t)\} = [\Psi_0]\{q(t)\} + [\Psi_1]\{\dot{q}(t)\} = [\Psi_s]\{z(t)\} \tag{12.79}$$

where $[\Psi_s] = \begin{bmatrix} [\Psi_0] & [\Psi_1] \end{bmatrix}$, and

$$[\Psi_0] = \begin{bmatrix} \{\beta_1\} & \{\beta_2\} & \cdots & \{\beta_r\} \end{bmatrix}^T, \quad [\Psi_1] = \begin{bmatrix} \{\gamma_1\} & \{\gamma_2\} & \cdots & \{\gamma_r\} \end{bmatrix}^T \tag{12.80}$$

Vectors $\{\beta_j\}$ and $\{\gamma_j\}$ are dependent upon the location and type of the jth sensor. These vectors are given in Table 12.4.2 for six types of sensors (S1 to S6). The output of the control system can be expressed as

$$\{y(t)\} = [g_s]\{y_s(t)\} = [C]\{z(t)\} \tag{12.81}$$

where $\{y(t)\}$ is a vector of output signals, $[g_s]$ is a constant gain matrix, and

$$[C] = [g_s][\Psi_s] \tag{12.82}$$

Feedback Controllers Three feedback control laws are considered:

(i) State feedback

$$\{U(t)\} = -[K_c]\{z(t)\} \tag{12.83}$$

where $[K_c]$ is a control gain matrix to be designed. The output equation (12.81) is not used in this control law. The closed-loop system is governed by

$$\{\dot{z}(t)\} = ([A] - [B][K_c])\{z(t)\} + \{B_e\} f_e(t) \tag{12.84}$$

which is obtained by substituting Eq. (12.83) into Eq. (12.76). The gain matrix is usually obtained by pole assignment or linear quadratic regulator (LQR) method.

(ii) Direct output feedback

$$\{U(t)\} = -[K_c]\{y(t)\} \tag{12.85}$$

where $\{y(t)\}$ is the vector of output signals given in Eq. (12.81), and $[K_c]$ is a control gain matrix to be designed. In this case, the closed-loop system is governed by

$$\{\dot{z}(t)\} = ([A] - [B][K_c][g_s][\Psi_s])\{z(t)\} + \{B_e\} f_e(t) \tag{12.86}$$

(iii) PID output feedback with one sensor and one actuator

Let the locations of the sensor and actuator be x_s and x_a, respectively. Under PID (proportional-integral-derivative) feedback, the control force in Eq. (12.59) is expressed as

$$F_c(x,t) = \begin{cases} u_c(t)\,\delta(x-x_a) & \text{for force actuator (A1)} \\ -u_c(t)\,\dfrac{d}{dx}\delta(x-x_a) & \text{for moment actuator (A2)} \end{cases} \tag{12.87}$$

with

$$u_c(t) = -\left(\frac{\partial}{\partial x}\right)^m \left(K_p w(x_s,t) + K_i \int_{t_0}^{t} w(x_s,\tau)\,d\tau + K_d \frac{\partial}{\partial t} w(x_s,t) \right) \tag{12.88}$$

where K_p, K_i, and K_d are the controller parameters to be designed; actuator types (A1) and (A2) are given in Table 12.4.1; $m = 0$, 1, and 2 for the sensors of types S1, S2, and S3 in Table 12.4.2, respectively. By the modal series (12.63), Eq. (12.59) is reduced to Eq. (12.76), with

$$[B_c] = \begin{cases} [\Phi(x_a)] & \text{for force actuator (A1)} \\ \dfrac{d}{dx}[\Phi(x_a)] & \text{for moment actuator (A2)} \end{cases} \tag{12.89}$$

and

$$\{U(t)\} = u_c(t) = -\{\lambda\}^T \left(K_p \{q(t)\} + K_i \int_{t_0}^{t} \{q(\tau)\}\,d\tau + K_d \{\dot{q}(t)\} \right) \tag{12.90}$$

where

$$\{\lambda\} = \begin{cases} [\Phi(x_s)] & \text{for displacement sensor (S1)} \\ \dfrac{d}{dx}[\Phi(x_s)] & \text{for rotation sensor (S2)} \\ \dfrac{d^2}{dx^2}[\Phi(x_s)] & \text{for strain sensor (S3)} \end{cases} \tag{12.91}$$

Define an auxiliary variable

$$\xi_a(t) = \{\lambda\}^T \int_{t_0}^{t} \{q(\tau)\}\,d\tau \tag{12.92}$$

such that the control force is

$$\{U(t)\} = -[K_c]\{z(t)\} - K_i\,\xi_a(t) \tag{12.93}$$

with

$$[K_c] = \begin{bmatrix} K_p \{\lambda\}^T & K_d \{\lambda\}^T \end{bmatrix} \tag{12.94}$$

Substituting Eq. (12.93) into Eq. (12.76) gives

$$\{\dot{z}(t)\} = ([A] - [B][K_c])\{z(t)\} - [B]K_i\xi_a(t) + \{B_e\}f_e(t) \tag{12.95}$$

Differentiating Eq. (12.92) gives

$$\dot{\xi}_a(t) = \{\lambda\}^T\{q(t)\} \tag{12.96}$$

Thus, combining Eqs. (12.95) and (12.96) yields an augmented state equation for the PID-controlled beam:

$$\begin{pmatrix}\{\dot{z}(t)\}\\ \dot{\xi}_a(t)\end{pmatrix} = \begin{bmatrix}[A] - [B][K_c] & -K_i[B]\\ \begin{bmatrix}\{\lambda\}^T & \{0\}^T\end{bmatrix} & 0\end{bmatrix}\begin{pmatrix}\{z(t)\}\\ \xi_a(t)\end{pmatrix} + \begin{pmatrix}\{B_e\}\\ 0\end{pmatrix}f_e(t) \tag{12.97}$$

It should be noted that the auxiliary variable $\xi_a(t)$ is needed only when integral feedback is involved in the control action. If $K_i = 0$, instead of Eq. (12.97), the following state equation is directly used:

$$\{\dot{z}(t)\} = ([A] - [B][K_c])\{z(t)\} + \{B_e\}f_e(t) \tag{12.98}$$

Response of Controlled Beam Once the control gain matrix $[K_c]$ or parameters K_p, K_i, K_d are selected, the response of the controlled beam can be determined numerically. To this end, write the closed-loop state equations (12.84), (12.86), and (12.97) in the unified form

$$\{\dot{\eta}(t)\} = [A_c]\{\eta(t)\} + \{B_a\}f_e(t) \tag{12.99}$$

where $\{\eta(t)\} = \{z(t)\}$ for state feedback or direct output feedback and $\{\eta(t)\} = \left(\{z(t)\}^T \quad \xi_a(t)\right)^T$ for PID feedback control; and $[A_c]$ is the associate state matrix. Rewrite Eq. (12.99) as

$$\{\dot{\eta}(t)\} = \Gamma(t, \{\eta(t)\}), \quad \{\eta(0)\} = \{\eta_0\} \tag{12.100}$$

where

$$\Gamma(t, \eta(t)) = [A_c]\{\eta(t)\} + \{B_a\}f_e(t) \tag{12.101}$$

The initial state is

$$\{\eta(0)\} = \begin{pmatrix}\{q(0)\}\\ \{\dot{q}(0)\}\end{pmatrix} \tag{12.102a}$$

for state or direct output feedback control, and

$$\{\eta(0)\} = \begin{pmatrix}\{q(0)\}\\ \{\dot{q}(0)\}\\ \xi_a(0)\end{pmatrix} \tag{12.102b}$$

for PID feedback control. Here

$$\begin{aligned}\{q(0)\} &= \begin{pmatrix} q_1(0) & q_2(0) & \cdots & q_n(0) \end{pmatrix}^T \\ \{\dot{q}(0)\} &= \begin{pmatrix} \dot{q}_1(0) & \dot{q}_2(0) & \cdots & \dot{q}_n(0) \end{pmatrix}^T \end{aligned} \tag{12.103}$$

with $q_k(t)$ being the modal coordinates of the beam. For PID control, zero initial value of the auxiliary variable can assumed; namely, $\xi_a(0) = 0$.

Equation (12.100) can be solved by a fixed-step Runge-Kutta algorithm of order four:

$$\{\eta_{k+1}\} = \{\eta_k\} + \frac{h}{6}\left(\{g_1\} + 2\{g_2\} + 2\{g_3\} + \{g_4\}\right) \tag{12.104}$$

where

$$\begin{aligned} \{g_1\} &= \Gamma\left(t_k, \{\eta_k\}\right) \\ \{g_2\} &= \Gamma\left(t_k + \frac{h}{2}, \{\eta_k\} + \frac{h}{2}\{g_1\}\right) \\ \{g_3\} &= \Gamma\left(t_k + \frac{h}{2}, \{\eta_k\} + \frac{h}{2}\{g_2\}\right) \\ \{g_4\} &= \Gamma\left(t_k + h, \{\eta_k\} + h\{g_3\}\right) \end{aligned} \tag{12.105}$$

$\{\eta_k\} = \{\eta(t_k)\}$, $t_k = kh$, $k = 0, 1, 2, \ldots$, and h is the step size in the numerical integration.

12.4.2 Solution by the Toolbox

The Toolbox of this chapter provides several MATLAB functions for modeling and simulation of Euler-Bernoulli beams under feedback control. The solution procedure by the Toolbox takes the following five steps:

Step 1. Set up beam parameters, boundary conditions, and number of modes considered in control system formulation by function `setbeam`;
Step 2. Obtain matrices $[A]$ and $[B]$ of the state equation (12.76) and matrix $[\Psi_s]$ of the sensor output equation (12.79) by function `ssbeam`;
Step 3. Design controller gain matrix $[K_c]$ or parameters K_p, K_i, K_d for one of the three control laws described in Section 12.4.1;
Step 4. With the designed controller parameters, obtain the matrices of the closed-loop state equation by function `clpbeam`; and
Step 5. Simulate the response of the controlled beam by function `contrbeam`.

The above solution procedure is illustrated in a flowchart in Fig. 12.4.3. (Also refer to Appendix B.7 for some commonly used MATLAB functions for control system design.) As can be seen, functions `ssbeam`, `clpbeam`, and `contrbeam` facilitate modeling and simulation of Euler-Bernoulli beams under feedback control. In addition, function `tfbeam` for constructing an open-loop transfer function for a beam may find use in controller design. These functions are presented in sequel.

Open-Loop Transfer Function Function `tfbeam` is used to obtain the open-loop transfer function $G_o(s)$ given in Eq. (12.72); see Window 4.1.

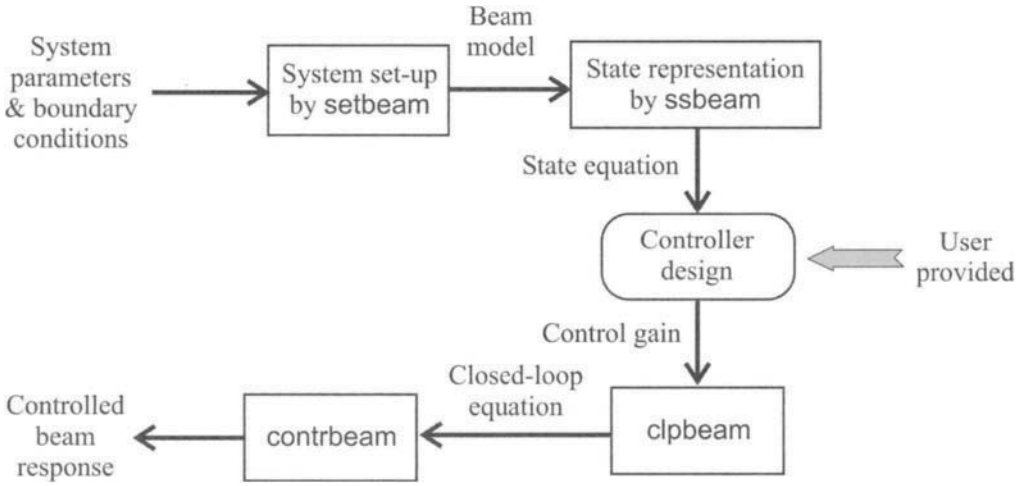


FIGURE 12.4.3 A flowchart of control system design and simulation by the Toolbox.

Window 4.1. Function `tfbeam`

MATLAB Function: **`tfbeam`**

Purpose:
To display the open-loop transfer function $G_o(s)$ of a beam given in Eq. (12.72).

Synopsis:

```
tfbeam(Sensor_Spec, Actuator_Spec)
[num, den, p, z, abc] = tfbeam(Sensor_Spec, Actuator_Spec)
```

Description:
`tfbeam(Sensor_Spec, Actuator_Spec)` displays the transfer function $G_o(s)$ and shows its poles and zeros in the complex plane. Here, `Sensor_Spec = [Type_no xs]` with `xs` being the sensor location and `Type_no` being an integer specifying the sensor type according to Table 12.4.2 as follows:

`Type_no = 1` displacement sensor (S1)
`Type_no = 2` slope sensor (S2)
`Type_no = 3` strain sensor (S3)

and `Sensor_Spec = [Type_no xa]` with `xa` being the actuator location and `Type_no` being an integer specifying the sensor type by Table 12.4.1 as follows:

`Type_no = 1` force actuator (A1)
`Type_no = 2` moment actuator (A2)

`[num, den, p, z, abc] = tfbeam(Sensor_Spec, Actuator_Spec)` returns the coefficients of the numerator and denominator of $G_o(s)$ in vectors `num` and `den`, respectively; the poles and zeros of $G_o(s)$ in vectors `p` and `z`, respectively; and the coefficients in the modal series (12.72) in a four-row matrix `abc`, with the first row storing the coefficients a_k, the second row b_k, the third row c_k, and the fourth row ω_k. With vectors `num` and `den`, transfer function $G_o(s)$ can be constructed by `tf(num, den)`, where `tf` is a function from the MATLAB Control System Toolbox.

EXAMPLE 4.1

In Fig. 12.4.4, a cantilever beam of linear density $\rho = 0.5$, bending stiffness $EI = 20$, and length $L = 1$ is controlled by a pair of sensor and actuator, where $u(t)$ is the control force applied to the beam by the actuator, and $y(t)$ is the output of the sensor. The sensor is of displacement type (S1), and is located at the beam tip ($x_s = L$). The actuator is of force type (A1) and is also located at the beam tip ($x_a = 1$). This is a colocated control system. The first four modes of vibration of the beam are considered in modeling and control. The commands

```
>> setbeam(0.5, 20, 1, [2 3], 0, 4)
>> tfbeam([1 1], [1 1])
```

yield the open-loop transfer function of the control system as follows

```
Transfer Function in Modal Series
---------------------------------
G(s) = Y(s)/U(s) = Term 1 + Term 2 + ··· + Term 4

where

            d1
Term 1 = ----------, with d1 = 8, w1 = 22.2372
         s^2 + w1^2

            d2
Term 2 = ----------, with d2 = 8, w2 = 139.3584
         s^2 + w2^2

            d3
Term 3 = ----------, with d3 = 8, w3 = 390.2074
         s^2 + w3^2

            d4
Term 4 = ----------, with d4 = 8, w4 = 764.6509
         s^2 + w4^2
```

Also, the transfer function poles and zeros are plotted in Fig. 12.4.5(a), which shows the alternation of the poles and zeros on the imaginary axis.

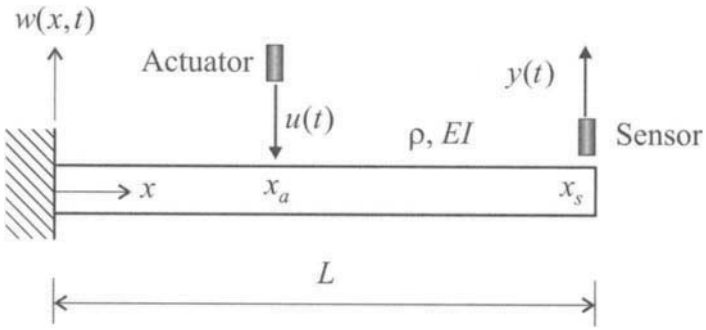


FIGURE 12.4.4

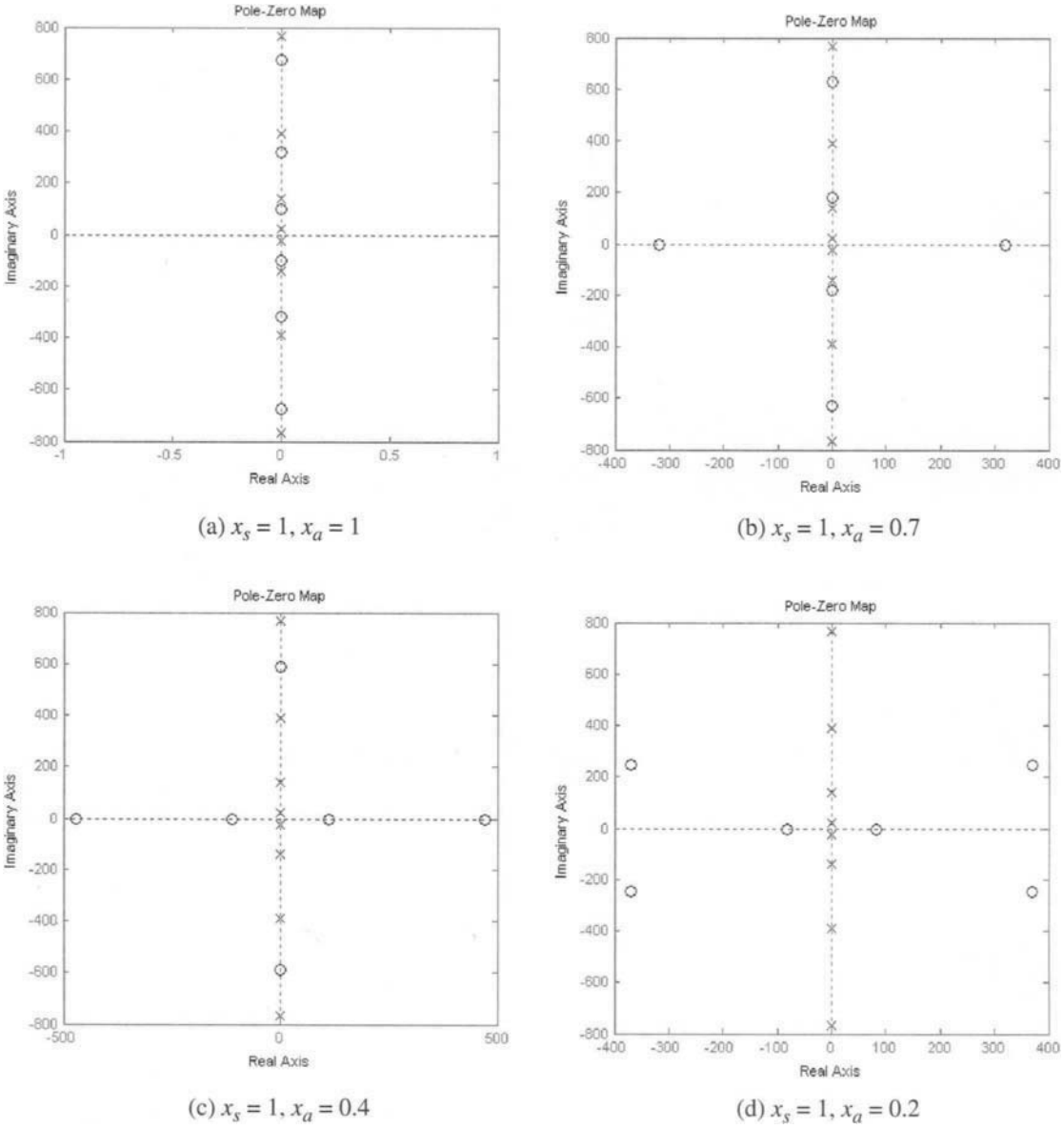


FIGURE 12.4.5 Transfer function poles and zeros.

Now, fix the sensor location at the beam tip and consider three actuator locations: $x_a = 0.7$, 0.4, and 0.2. By

```
tfbeam([1 1], [1 0.7])  % xa = 0.7
tfbeam([1 1], [1 0.4])  % xa = 0.4
tfbeam([1 1], [1 0.2])  % xa = 0.2
```

the poles and zeros of the open-loop transfer function at the new actuator locations are plotted in Figs. 12.4.5(b), (c), and (d), respectively. Because the actuator is not at the

beam tip, the control system becomes noncolocated. As the actuator moves toward the clamped end of the beam, the zeros on the imaginary axis gradually disappear, and in the meantime, show up in the complex plane.

Note that in Figs. 12.4.5(b), (c), and (d), there are zeros in the right-half complex plane. A system with zero(s) in the open right-half plane is known as a non-minimum-phase system, while a system with all zeros in the closed left-half plane (including the imaginary axis) is known as a minimum-phase system. To see the meaning of "minimum phase," compare the bode diagrams of the open-loop transfer function in Figs. 12.4.6

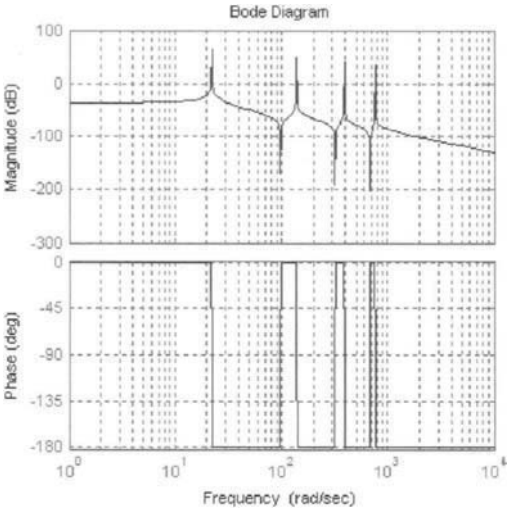


FIGURE 12.4.6 $x_s = x_a = 1$.

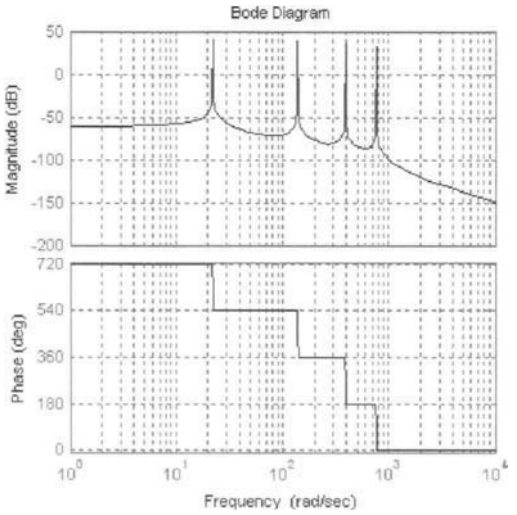


FIGURE 12.4.7 $x_s = 1$, $x_a = 0.2$.

and 12.4.7 for colocated sensor and actuator ($x_s = x_a = 1$) and noncolocated sensor and actuator ($x_s = 1, x_a = 0.2$), respectively. These figures are obtained by

```
% Fig. 12.4.6 colocated sensor and actuator
setbeam(0.5, 20, 1, [2 3], 0.001, 4)
[num1, den1] = tfbeam([1 1], [1 1]);
bode(num1, den1); grid
% Fig. 12.4.7 noncolocated sensor and actuator
[num2, den2] = tfbeam([1 1], [1 0.2]);
bode(num2, den2); grid
```

Here, for smooth graphs, a small viscous damping ($d_v = 0.001$) has been introduced. The phase plot in Fig. 12.4.6 is bounded from below by a minimum phase (-180 degrees in this case). The phase curve in Fig. 12.4.7 keeps going down as the frequency increases, and it does not have such a minimum phase feature.

State Representation To obtain the state equation (12.76) and the sensor output equation (12.79) function `ssbeam` is used; see Window 4.2.

Window 4.2. Function ssbeam

MATLAB Function: **ssbeam**

Purpose:
To obtain a state representation for a beam.

Synopsis:

```
ssbeam
A = ssbeam
ssbeam(Sensor_Spec, Actuator_Spec)
[A, B, PHI_S] = ssbeam(Sensor_Spec, Actuator_Spec)
```

Description:
`ssbeam` obtains the state matrix [A] in Eq. (12.76) without introducing feedback control.

`A = ssbeam` returns the state matrix in `A`.

`ssbeam(Sensor_Spec, Actuator_Spec)` obtains a state representation for a beam under feedback control with r sensors and s actuators, which is described by Eqs. (12.76) and (12.79). Here, `Sensor_Spec` is an r-row matrix that describes the sensors by

$$\texttt{Sensor\_Spec} = \begin{bmatrix} \texttt{Sensor\_1} \\ \texttt{Sensor\_2} \\ \vdots \\ \texttt{Sensor\_r} \end{bmatrix}$$

Window 4.2. Function ssbeam (*Continued*)

where the jth row `Sensor_j` specifies the type and location of the jth sensor according to Table 12.4.2; and `Actuator_Spec` is an s-row matrix that describes the actuators by

$$\texttt{Actuator\_Spec} = \begin{bmatrix} \texttt{Actuator\_1} \\ \texttt{Actuator\_2} \\ \vdots \\ \texttt{Actuator\_s} \end{bmatrix}$$

where the jth row `Actuator_j` specifies the type and location of the jth actuator according to Table 12.4.1.

`[A, B, PHI_S] = ssbeam(Sensor_Spec, Actuator_Spec)` returns the matrices $[A]$ and $[B]$ of the state equation (12.76) in `A` and `B`, and the matrix $[\Psi_s]$ in the sensor output equation (12.79) in `PHI_S`.

Note:

Before function `ssbeam` can be used, `setbeam` must be executed.

As listed in Tables 12.4.1 and 12.4.2, two types of actuators and six types of sensors are considered in this Toolbox. In specification of `Sensor_Spec` and `Actuator_Spec`, if needed, more than one type of sensor actuator can be placed at one point. Also, sensors and actuators can be placed at the same point as well. Once the matrices of the state equation are obtained, the MATLAB Control System Toolbox can be used to examine the properties of the control system (e.g., stability, controllability, and observability) and to design a feedback controller.

EXAMPLE 4.2

In Fig. 12.4.8, a clamped-pinned beam ($\rho = 1, EI = 10, L = 1$) is under a feedback controller, which has a sensor at $x_s = 0.65$ and an actuator at $x_a = 0.65$. The sensor measures the transverse displacement $y(t) = w(x_s, t)$; the actuator applies a transverse force $u(t)$ to the beam. This is a single-input-single-output (SISO) system.

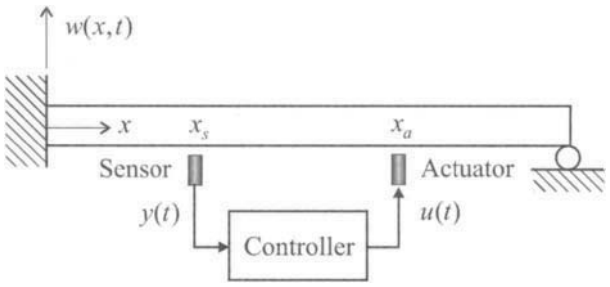


FIGURE 12.4.8

The first three modes of vibration are considered in the control system formulation. The matrices in Eqs. (12.76) and (12.79) are obtained by

```
>> setbeam(1, 10, 1, [2 1], 0, 3)
>> xs = 0.65; xa = 0.65;
>> [A, B, C] = ssbeam([1 xs], [1 xa]);
```

which yields

```
Matrix [A]:
       0       0       0   1.0000e+000       0       0
       0       0       0       0   1.0000e+000       0
       0       0       0       0       0   1.0000e+000
-2.3772e+003       0       0       0       0       0
       0 -2.4965e+004       0       0       0       0
       0       0 -1.0868e+005       0       0       0

Matrix [B]:
     0
     0
     0
1.4605e+000
8.6550e-001
-5.9076e-001

Matrix [PHI_S]:
 1.4605e+000 8.6550e-001 -5.9076e-001       0       0       0
```

The transfer function of the beam from the control force to the sensor output is

$$G_b(s) = \frac{Y(s)}{U(s)} = [C]\,(s\,[I] - [A])^{-1}\,[B]$$

where $Y(s)$ and $U(s)$ are the Laplace transforms of $w(x_s, t)$ and $u(t)$, respectively, and $[C] = [\Psi_s]$. The transfer function, of course, can also be expressed in a modal series by function `tfbeam` (Window 4.1). Consider a PID control law

$$\frac{U(s)}{Y(s)} = -\mu(s+2)$$

where μ is a positive gain parameter. The closed-loop characteristic equation is

$$1 + \mu\,(s+2)\,G_b(s) = 0.$$

The root loci of the closed-loop system as μ varies from zero to infinity are plotted in Fig. 12.4.9 by

```
>> [num, den] = ss2tf(A,B,C,0);
>> Gb = tf(num, den);
>> Gc = tf([1 2], 1);
>> G_open = series(Gc,Gb);
>> rlocus(G_open)
>> axis([-60 10 -350 350])
```

where `ss2tf`, `tf`, and `rlocus` are functions from the MATLAB Control System Toolbox (see Appendix B.7). As can be seen from Fig. 12.4.9, the beam under the PID control is stable for any $\mu > 0$. Note that the sensor and actuator are colocated ($x_s = x_a = 0.65$). If the sensor and actuator are noncolocated, say $x_s = 0.5$ and $x_a = 0.65$, the root loci of the corresponding closed-loop are plotted in Fig. 12.4.10, which indicates instability caused by noncolocation of the sensor and actuator.

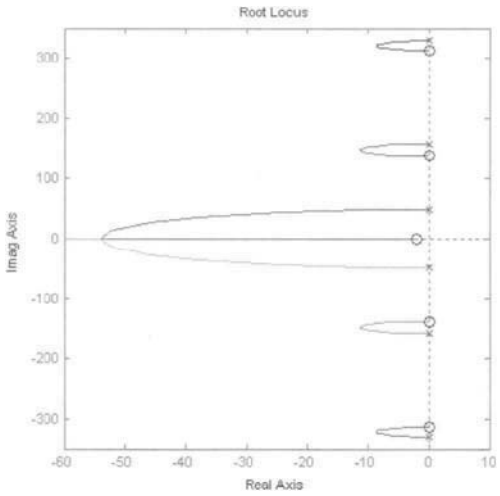


FIGURE 12.4.9 Colocated control.

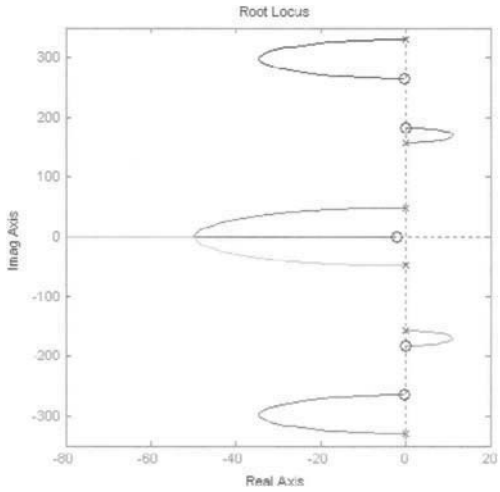


FIGURE 12.4.10 Noncolocated control.

Closed-Loop The closed-loop state equation of a beam under feedback control is obtained by function `clpbeam`; see Window 4.3.

Window 4.3. Function clpbeam

MATLAB Function: **clpbeam**

Purpose:
To obtain the closed-loop state equation for a beam under feedback control.

Synopsis:

```
clpbeam(ID_control, K_gain, Gs)
Ac = clpbeam(ID_control, K_gain, Gs)
```

Description:
`clpbeam(ID_control, K_gain, Gs)` obtains the closed-loop state equation for a beam under feedback control, and computes the closed-loop eigenvalues. Here, `ID_control` is an integer identifying one of the three feedback controllers given in Section 12.4.1, and `K_gain` and `Gs` are gain matrices as explained below.

`ID_control` = 1: state feedback $\{U(t)\} = -[K_c]\{z(t)\}$. The closed-loop system is described by Eq. (12.84). `K_gain` is the control gain matrix $[K_c]$. `Gs` is not assigned. In this case, `clpbeam(ID_control, K_gain)` is adequate.

`ID_control` = 2: output feedback $\{U(t)\} = -[K_c]\{y(t)\}$. The closed-loop system is given by Eq. (12.86). `K_gain` is the control gain matrix $[K_c]$, and `Gs` is matrix $[g_s]$ in Eq. (12.81). When `Gs` is an identify matrix, i.e., $\{y(t)\} = \{y_s(t)\}$ by Eq. (12.81), `clpbeam(ID_control, K_gain)` is used without having to assign `Gs`.

`ID_control` = 3: PID feedback given by Eq. (12.88). The closed-loop system is described by the augmented equation (12.97). In this case, `K_gain = [Kp Ki Kd]`, where `Kp`, `Ki`, and `Kd` are the controller parameters, and `Gs` is not assigned.

`ID_control` = 0: no control action, $\{U(t)\} = 0$. In this case, there is no need to assign `K_gain` and `Gs`, and `clpbeam(0)` can be directly used.

`Ac = clpbeam(ID_control, K_gain, Gs)` returns the state matrix of the closed-loop system in `Ac`.

Note:

Before `clpbeam` can be used, function `ssbeam` should be executed.

EXAMPLE 4.3

Consider the same beam as given in Example 4.2. A displacement sensor and a force actuator are colocated at $x_s = x_a = 0.65$. Consider the PID feedback

$$F_c(x,t) = -\left(K_p w(x_s,t) + K_i \int_{t_0}^{t} w(x_s,\tau)\,d\tau + K_d \frac{\partial}{\partial t} w(x_s,t)\right) \delta(x - x_a)$$

with gains $K_p = 12$, $K_i = 6$, and $K_d = 5$. The commands

```
>> setbeam(1, 10, 1, [2 1], 0, 3)
>> ssbeam([1 0.65], [1 0.65])
>> clpbeam(3, [12 6 5])
```

give the state matrix in the augmented state equation (12.97)

```
Matrix [Ac]:
      0       0       0 1.0000e+000       0       0       0
      0       0       0       0 1.0000e+000       0       0
      0       0       0       0       0 1.0000e+000       0
-2.4028e+003 -1.5169e+001 1.0354e+001 -1.0665e+001 -6.3202e+000 4.3140e+000 -8.7629e+000
-1.5169e+001 -2.4974e+004 6.1357e+000 -6.3202e+000 -3.7454e+000 2.5565e+000 -5.1930e+000
 1.0354e+001  6.1357e+000 -1.0868e+005 4.3140e+000  2.5565e+000 -1.7450e+000  3.5446e+000
 1.4605e+000  8.6550e-001 -5.9076e-001      0        0       0       0
```

and the closed-loop eigenvalues

```
Eigenvalues of [Ac]
-8.7113e-001 +3.2962e+002i
-8.7113e-001 -3.2962e+002i
-1.8677e+000 +1.5789e+002i
-1.8677e+000 -1.5789e+002i
-5.3362e+000 +4.8774e+001i
-5.3362e+000 -4.8774e+001i
-5.5215e-003
```

Simulation of Controlled Beam Response Once the closed-loop state equation is obtained, the response of the controlled beam can be computed by function `contrbeam`; see Window 4.4.

Window 4.4. Function `contrbeam`

MATLAB Function: **`contrbeam`**

Purpose:
To compute the response of a beam with or without feedback control, by the Runge-Kutta algorithm described in Eqs. (12.104) and (12.105).

Synopsis:

```
contrbeam(xr, Load_Spec, IC, tf, npts)
[w, t, u, z] = contrbeam(xr, Load_Spec, IC, tf, npts)
```

Description:
`contrbeam(xr, Load_Spec, IC, tf, npts)` plots the displacement of a beam (with or without feedback control) at `xr` versus time, where `Load_Spec` is a vector specifying an external load by Table 12.3.2, `IC` is a two-row matrix specifying the initial displacement and velocity of the beam according to Window 3.3, `tf` is total time of simulation, and `npts` the number of equally-spaced points in the time region `[0, tf]`. By default, `tf` is about four times the first natural period of the beam, and `npts` = 1,001. If zero external force or zero initial disturbances are assigned, `Load_Spec` or `IC` is replaced by 0 or [ ]. Also, `tf` can be replaced by [ ] as a placeholder for its default value. For instance, for a beam subject to an external force only, one can use `contrbeam(xr, Load_Spec)`.

Window 4.4. Function contrbeam (*Continued*)

`[w, t, u, z] = contrbeam(xr, Load_Spec, IC, tf, npts)` returns the time history of the beam response at `xr` in a two-row matrix `w`, the times of simulation in a vector `t`, the time history of the control forces by the actuators in a matrix `u`, and the time history of the state vector in a matrix `z`. With `w` and `t`, the beam response versus time can be plotted in one of the following two ways:

(i) Use the MATLAB function `plot` as follows:

```
plot(t, w(1,:))   for displacement
plot(t, w(2,:))   for velocity
```

(ii) Use function `plotbeam(w, opt)` as shown in Window 3.6.

Similarly, the force by the kth actuator versus time can be plotted by `plot(t, u(k,:))`; the time history of the jth state variable can be plotted by `plot(t, z(j,:))`.

Note:

Before function `contrbeam` can be used, `ssbeam` (for a beam with no control) or `clpbeam` (for a beam with feedback control) must be executed. Also, for an uncontrolled beam, function `contrbeam` provides a numerical alternative for solution of Eq. (12.25), compared to the modal expansion implemented by functions `freebeam` and `forcedbeam`.

The input arguments `tf` and `npts` of function `contrbeam` set up the step size h of the Runge-Kutta algorithm in Eq. (12.104) as follows

$$h = \frac{\texttt{tf}}{\texttt{npts-1}}$$

For accuracy in numerical simulation, it is suggested that $h \leq \pi/(3\omega_n)$, where ω_n is the highest natural frequency of the beam model that is given by the n-term modal series (12.63).

EXAMPLE 4.4

Consider a clamped-free (cantilever) beam of parameters $\rho = 0.5$, $EI = 18$, and $L = 2.4$. The beam has 5% modal damping in each mode. A pointwise step load of unit amplitude is applied to the beam at $x_f = 1.6$. Assume zero initial disturbances. In modeling of the beam dynamic response, the first four modes of vibration are considered. Without control, the beam displacement at its tip ($x_r = 2.4$) is plotted in Fig. 12.4.11 by

```
>> setbeam(0.5, 18, 2.4, [2 3], [1 0.05], 4)
>> ssbeam
>> [y0, t0] = forcedbeam([1 1.6 1], 2.4, 4, 101);
>> [y, t] = contrbeam(2.4, [1 1.6 1], 0, 4, 1001);
>> clf
>> plot(t0, y0(1,:),'o', t, y(1,:)); legend('Modal analysis', 'RK
solution')
>> xlabel('Time t'); title('Beam displacement at xr = 2.4')
```

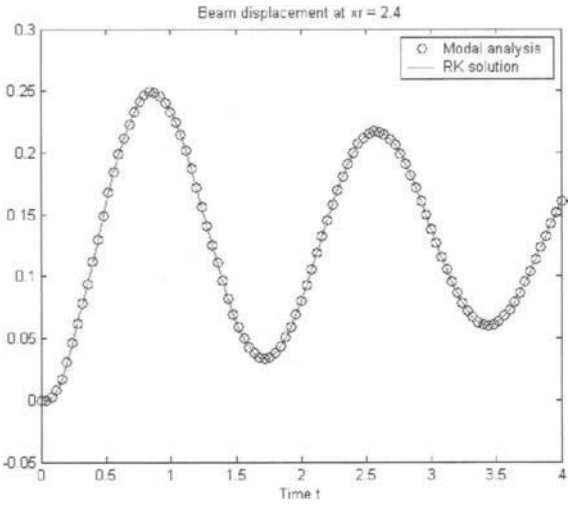


FIGURE 12.4.11

where the step size of the Runge-Kutta algorithm is $h = 0.004$. For comparison purpose, function `forcedbeam` has been used to give the modal expansion solution. Define the error of the Runge-Kutta (RK) solution by

$$\text{Error} = |w(x_r, t) - w_{RK}(x_r, t)|$$

where $w(x_r, t)$ is the solution obtained by `forcedbeam`, and $w_{RK}(x_r, t)$ is the solution by `contrbeam`. The error is plotted against time in Fig. 12.4.12, which is less than 3×10^{-7} initially, and decays to zero as time goes by.

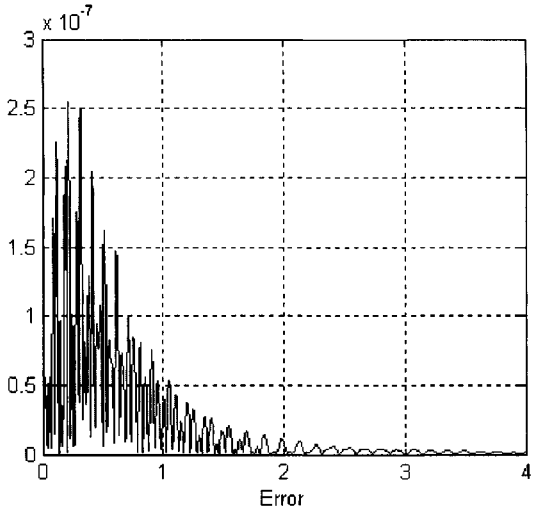


FIGURE 12.4.12

Now, consider that the beam is under PID feedback control, with one displacement sensor and one force actuator that are colocated at $x = 1.2$. The controller transfer function is

$$G_c(s) = \frac{Y(s)}{U(s)} = -(K_p + K_d s)$$

where $Y(s)$ is the Laplace transform of the beam response at the sensor location and $U(s)$ is the Laplace transform of the control force by the actuator. Let the controller gains be $K_p = 5$ and $K_d = 10$. The tip displacement ($x_r = 2.4$) of the beam with the PID feedback control and without control is plotted in Fig. 12.4.13 by

```
>> ssbeam([1 1.2], [1 1.2]);
>> clpbeam(3, [5 0 10]);
>> [yc, t, u] = contrbeam(2.4, [1 1.6 1], 0, 4, 1001);
>> clf
>> plot(t, y(1,:), t, yc(1,:),':'); legend('No control', 'PD
control')
>> xlabel('Time t'); title('Beam displacement at xr = 2.4')
```

In addition,

```
>> plot(t, yc(1,:), t, yc(2,:), ':', t, u, '-.');
>> legend('Displacement', 'Velocity', 'Control force')
>> xlabel('Time t')
```

plots the time history of the displacement and velocity of the controlled beam at the tip, and that of the control force; see Fig. 12.4.14.

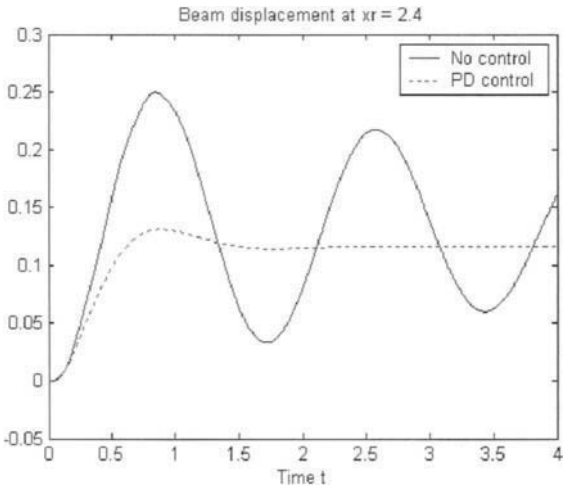


FIGURE 12.4.13

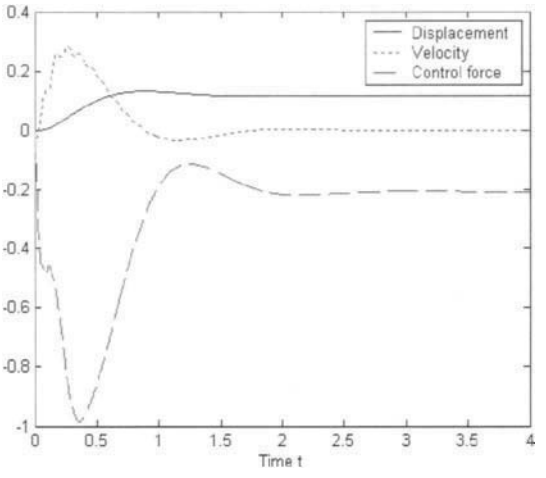


FIGURE 12.4.14

12.5 Dynamics and Control of Nonuniform Beams

The previous sections are concerned with uniform beams. In this section, vibration and control of Euler-Bernoulli beams with nonuniform linear density and bending stiffness is addressed; the corresponding MATLAB functions are presented.

12.5.1 Problem Statement

Shown in Fig. 12.5.1 is a nonuniform Euler-Bernoulli beam with its linear density (mass per unit length) $\rho(x)$ and bending stiffness $EI(x)$ as functions of the spatial coordinate x. The transverse displacement $w(x,t)$ of the beam is governed by the differential equation

$$\rho(x)\frac{\partial^2}{\partial t^2}w(x,t)+\frac{\partial^2}{\partial x^2}\left[EI(x)\frac{\partial^2}{\partial x^2}w(x,t)\right]=F(x,t),\quad 0\le x\le L \tag{12.106}$$

along with the boundary conditions

$$B_{Lj}\left[w(x,t)\right]_{x=0}=0,\quad B_{Rj}\left[w(x,t)\right]_{x=L}=0,\quad j=1,2 \tag{12.107}$$

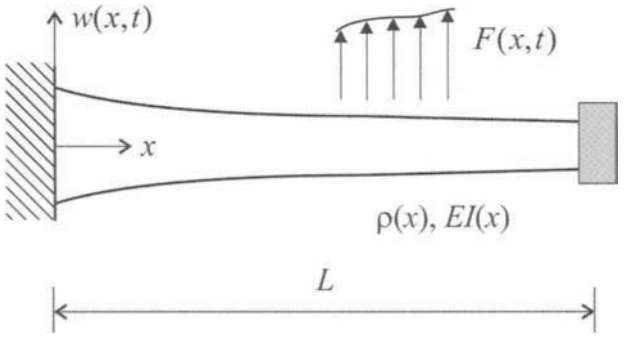


FIGURE 12.5.1 Schematic of a nonuniform Euler-Bernoulli beam.

and the initial conditions

$$w(x,0) = a_0(x), \quad \frac{\partial}{\partial t}w(x,0) = b_0(x); \quad 0 \le x \le L \tag{12.108}$$

where $F(x,t)$ is an external force; B_{Lj} and B_{Rj} are temporal-spatial differential operators; and $a_0(x)$ and $b_0(x)$ are the distributions of initial displacement and velocity of the beam. The rotation (slope), bending moment, and shear force of the beam are

$$\theta(x,t) = \frac{\partial}{\partial x}w(x,t), \quad M(x,t) = EI(x)\frac{\partial^2}{\partial x^2}w(x,t), \quad Q(x,t) = \frac{\partial}{\partial x}\left[EI(x)\frac{\partial^2}{\partial x^2}w(x,t)\right] \tag{12.109}$$

Listed in Table 12.5.1 are 10 types of boundary conditions for nonuniform beams' where k_r is the coefficient of a torsional spring, k_t is the coefficient of a translational spring, m and I are the inertia and moment of inertia of an end mass, and I_D is the moment of inertia of a hinged rigid disk. Compared to Table 12.2.1 for uniform beams, Table 12.5.1 has three additional types of boundary conditions (B8) to (B10), which are related to end inertia (lumped mass or rigid disk) at beam boundaries.

A fundamental problem in dynamic analysis of nonuniform Euler-Bernoulli beams is to solve the boundary-initial value problem described by Eqs. (12.106) to (12.108). Due to the nonuniform distribution of linear density and bending stiffness, exact solution of such a problem is limited to a few special cases. In general, approximate solution methods have to be used. The subsequent sections present an approximate solution method for nonuniform Euler-Bernoulli beams.

12.5.2 Rayleigh-Ritz Discretization

Many approximate methods are available for modeling and dynamic analysis of general structural systems, among which are finite element methods, finite difference methods, and Rayleigh-Ritz methods. This section presents a Rayleigh-Ritz method for the boundary-initial value problem discussed in Section 12.5.1. The reader who is only interested in using the Toolbox of this chapter may skip this section and move to subsequent sections.

The basic idea behind the Rayleigh-Ritz method is to approximate the displacement of a nonuniform beam by an N-term series

$$w(x,t) \approx w_N(x,t) = \sum_{k=1}^{N} \phi_k(x) q_k(t) \tag{12.110}$$

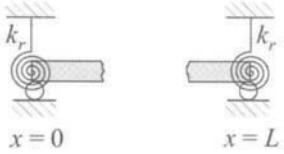
where $\phi_k(x)$ are from an infinite sequence of *admissible functions* and $q_k(t)$ are unknown generalized coordinates. The method is also referred to as *assumed-modes method* (Reference 4). The admissible functions of a beam are those that are differentiable and satisfy all geometric boundary conditions of the beam. Geometric boundary conditions are those that only involve the displacement $w(x,t)$ and rotation $\frac{\partial}{\partial x}w(x,t)$. (Boundary conditions involving higher derivatives $\frac{\partial^2}{\partial x^2}w(x,t)$ and $\frac{\partial^3}{\partial x^3}w(x,t)$ are called *force boundary conditions*.) Three examples of geometric boundary conditions are given as follows:

(a) Simply-supported (pinned-pinned) beam: $w(0,t) = 0$, $w(L,t) = 0$.

(b) Cantilever (clamped-free) beam: $w(0,t) = 0$, $\frac{\partial w(0,t)}{\partial x} = 0$.

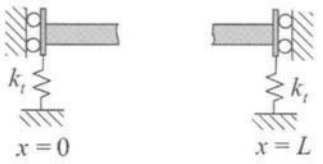
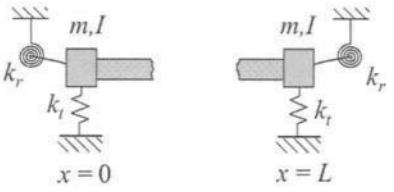
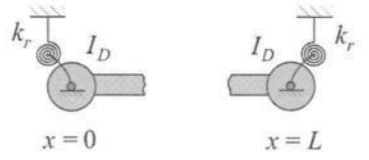
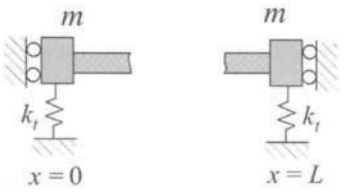
(c) Free-free beam: no geometric boundary conditions.

TABLE 12.5.1 Boundary Conditions of Nonuniform Beams in Transverse Vibration

Boundary Type	*Boundary Conditions*
(B1) Pinned or hinged end	Left end: $w(0,t) = 0, \quad -EI(0)\dfrac{\partial^2 w(0,t)}{\partial x^2} = 0$ Right end: $w(L,t) = 0, \quad EI(L)\dfrac{\partial^2 w(L,t)}{\partial x^2} = 0$
(B2) Clamped or fixed end	Left end: $w(0,t) = 0, \quad \dfrac{\partial w(0,t)}{\partial x} = 0$ Right end: $w(L,t) = 0, \quad \dfrac{\partial w(L,t)}{\partial x} = 0$
(B3) Free end	Left end: $-EI(0)\dfrac{\partial^2 w(0,t)}{\partial x^2} = 0, \quad \dfrac{\partial}{\partial x}\left[EI(x)\dfrac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=0} = 0$ Right end: $EI(L)\dfrac{\partial^2 w(L,t)}{\partial x^2} = 0, \quad -\dfrac{\partial}{\partial x}\left[EI(x)\dfrac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=L} = 0$
(B4) Sliding or guided end	Left end: $\dfrac{\partial w(0,t)}{\partial x} = 0, \quad \dfrac{\partial}{\partial x}\left[EI(x)\dfrac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=0} = 0$ Right end: $\dfrac{\partial w(L,t)}{\partial x} = 0, \quad -\dfrac{\partial}{\partial x}\left[EI(x)\dfrac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=L} = 0$
(B5) Elastic-support	Left end: $k_r\dfrac{\partial w(0,t)}{\partial x} - EI(0)\dfrac{\partial^2 w(0,t)}{\partial x^2} = 0$ $k_t w(0,t) + \dfrac{\partial}{\partial x}\left[EI(x)\dfrac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=0} = 0$ Right end: $k_r\dfrac{\partial w(L,t)}{\partial x} + EI(L)\dfrac{\partial^2 w(L,t)}{\partial x^2} = 0$ $k_t w(L,t) - \dfrac{\partial}{\partial x}\left[EI(x)\dfrac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=L} = 0$
(B6) Hinged-elastic end 	Left end: $w(0,t) = 0, \quad k_r\dfrac{\partial w(0,t)}{\partial x} - EI(0)\dfrac{\partial^2 w(0,t)}{\partial x^2} = 0$ Right end: $w(L,t) = 0, \quad k_r\dfrac{\partial w(L,t)}{\partial x} + EI(L)\dfrac{\partial^2 w(L,t)}{\partial x^2} = 0$

(*continued*)

TABLE 12.5.1 Boundary Conditions of Nonuniform Beams in Transverse Vibration (*Continued*)

Boundary Type	*Boundary Conditions*
(B7) Sliding-elastic end 	Left end: $\frac{\partial w(0,t)}{\partial x}=0,\quad k_t w(0,t)+\frac{\partial}{\partial x}\left[EI(x)\frac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=0}=0$ Right end: $\frac{\partial w(L,t)}{\partial x}=0,\quad k_t w(L,t)-\frac{\partial}{\partial x}\left[EI(x)\frac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=L}=0$
(B8) End mass 	Left end: $I\frac{\partial^3 w(0,t)}{\partial t^2\partial x}+k_r\frac{\partial w(0,t)}{\partial x}-EI(0)\frac{\partial^2 w(0,t)}{\partial x^2}=0$ $m\frac{\partial^2 w(0,t)}{\partial t^2}+k_t w(0,t)+\frac{\partial}{\partial x}\left[EI(x)\frac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=0}=0$ Right end: $I\frac{\partial^3 w(L,t)}{\partial t^2\partial x}+k_r\frac{\partial w(L,t)}{\partial x}+EI(L)\frac{\partial^2 w(L,t)}{\partial x^2}=0$ $m\frac{\partial^2 w(L,t)}{\partial t^2}+k_t w(L,t)-\frac{\partial}{\partial x}\left[EI(x)\frac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=L}=0$
(B9) Hinged disk 	Left end: $w(0,t)=0,\ I_D\frac{\partial^3 w(0,t)}{\partial t^2\partial x}+k_r\frac{\partial w(0,t)}{\partial x}-EI(0)\frac{\partial^2 w(0,t)}{\partial x^2}=0$ Right end: $w(L,t)=0,\ I_D\frac{\partial^3 w(L,t)}{\partial t^2\partial x}+k_r\frac{\partial w(L,t)}{\partial x}+EI(L)\frac{\partial^2 w(L,t)}{\partial x^2}=0$
(B10) Sliding mass 	Left end: $\frac{\partial w(0,t)}{\partial x}=0$ $m\frac{\partial^2 w(0,t)}{\partial t^2}+k_t w(0,t)+\frac{\partial}{\partial x}\left[EI(x)\frac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=0}=0$ Right end: $\frac{\partial w(L,t)}{\partial x}=0$ $m\frac{\partial^2 w(L,t)}{\partial t^2}+k_t w(L,t)-\frac{\partial}{\partial x}\left[EI(x)\frac{\partial^2 w(x,t)}{\partial x^2}\right]_{x=L}=0$

Solution of the boundary-initial value problem by $w_N(x,t)$ takes the following three steps:

Step 1. Express the kinetic energy and strain energy of the beam in terms of $w_N(x,t)$. Because $\phi_k(x)$ are known, the kinetic energy and strain energy become functions of $q_k(t)$.

Step 2. Derive a set of ordinary differential equations governing the general coordinates $q_k(t)$ by generalized Hamilton's principle or Lagrange's equations.

Step 3. Solve the ordinary differential equations for $q_k(t)$, which, by Eq. (12.110), give the approximate response of the nonuniform beam.

The above solution procedure is also known as *discretization process* because it reduces an infinite-dimensional system described by the partial differential equation (12.106) to a

finite-dimensional one governed by a set of ordinary differential equations. Mathematically, if the infinite sequence $\{\phi_k\}_{k=1}^{\infty}$ forms a complete basis in the function space of the beam problem, $w_N(x,t)$ approaches the exact solution of Eqs. (12.106) to (12.108) as the number N goes to infinity.

The above discretization is detailed in sequel.

Kinetic Energy and Strain Energy The kinetic energy of a nonuniform beam is written as

$$T = \frac{1}{2}\int_0^L \rho(x)\left(\frac{\partial w(x,t)}{\partial t}\right)^2 dx + T_b \tag{12.111}$$

where T_b accounts for the kinetic energy of end inertia. For instance, for a beam with a lumped mass (B8 in Table 12.5.1) at its left end, and a hinged rigid disk (B9) at its right end

$$T_b = \frac{1}{2}m\left(\frac{\partial w(0,t)}{\partial t}\right)^2 + \frac{1}{2}I\left(\frac{\partial^2 w(0,t)}{\partial t \partial x}\right)^2 + \frac{1}{2}I_D\left(\frac{\partial^2 w(L,t)}{\partial t \partial x}\right)^2.$$

According to Table 12.5.1, T_b only needs to be added to Eq. (12.111) when a beam has boundary conditions B8 to B10. For the beam with boundary conditions B1 to B7, $T_b = 0$.

The strain energy of a nonuniform beam is of the form

$$U = \frac{1}{2}\int_0^L EI(x)\left(\frac{\partial^2 w(x,t)}{\partial x^2}\right)^2 dx + U_b \tag{12.112}$$

where U_b is the potential energy of end springs. For example, for a beam pinned at its left end and elastically supported (B5) at its right end,

$$U_b = \frac{1}{2}k_r\left(\frac{\partial w(L,t)}{\partial x}\right)^2 + \frac{1}{2}k_t w^2(L,t).$$

From Table 12.5.1, U_b is only considered for boundary conditions B5 to B10, and is zero for boundary conditions B1 to B4.

Admissible Functions Given a nonuniform beam, define a *nominal beam* by the following two conditions:

(i) The nominal beam is uniformly distributed, with its linear density and bending stiffness as the averages of the parameters of the nonuniform beam:

$$\bar{\rho} = \frac{1}{L}\int_0^L \rho(x)\,dx, \quad \overline{EI} = \frac{1}{L}\int_0^L EI(x)\,dx \tag{12.113}$$

(ii) The boundary conditions of the nominal beam include all the geometric boundary conditions of the nonuniform beam. In particular, if the nonuniform beam has boundary conditions B1 to B7 as listed in Table 12.5.1, the nominal beam has the similar boundary conditions as given Table 12.2.1. If the nonuniform beam has boundary conditions B8, B9 or B10, the nominal beam is assigned boundary conditions B5, B6 or B7, respectively.

Consider the eigenvalue problem of the so-defined nominal beam

$$\bar{\rho}\alpha_k^2\phi_k(x) = \overline{EI}\frac{d^4}{dx^4}\phi_k(x), \quad k = 1, 2, \ldots \tag{12.114}$$

where α_k are the natural frequencies of the nominal beam and ϕ_k are the associate mode shapes. As shown in Section 12.2.5, the exact eigensolutions of Eq. (12.114) can be obtained by the Distributed Transfer Function Method, and MATLAB function `setbeam` from the Toolbox is available for such solutions.

The eigenfunctions ϕ_k are differentiable and satisfy all geometric boundary conditions of the nonuniform beam. Thus, ϕ_k can be chosen as the admissible functions in the series (12.110). Also, it is known that the sequence $\{\phi_k\}_{k=1}^{\infty}$ forms a complete basis in the function space of the beam problem, which guarantees the convergences of the solution $W_N(x, t)$.

Discretization With the eigenfunctions of the nonminal beam chosen as admissible functions, a discretized model of the nonuniform beam is derivable. To this end, substitute the series (12.110) into the energy expressions (12.111) and (12.112), to obtain

$$\begin{aligned} T &= \frac{1}{2}\{\dot{q}(t)\}^T [M]\{\dot{q}(t)\} \\ U &= \frac{1}{2}\{q(t)\}^T [K]\{q(t)\} \end{aligned} \tag{12.115}$$

where $\{q(t)\}$ is the vector of generalized coordinates, i.e.,

$$\{q(t)\} = (q_1(t) \quad q_2(t) \quad \cdots \quad q_N(t))^T \tag{12.116}$$

and $[M]$ and $[K]$ are the mass and stiffness matrices of the discretized model of the nonuniform beam. Write

$$[M] = [M_{ij}], \quad [K] = [K_{ij}] \tag{12.117}$$

The elements of the mass and stiffness matrices are given by

$$\begin{aligned} M_{ij} &= \int_0^L \rho(x)\phi_i(x)\phi_j(x)\,dx + \alpha_{ij}^0 + \alpha_{ij}^L \\ K_{ij} &= \int_0^L EI(x)\frac{d^2\phi_i(x)}{dx^2}\frac{d^2\phi_j(x)}{dx^2}\,dx + \beta_{ij}^0 + \beta_{ij}^L \end{aligned} \tag{12.118}$$

where ϕ_k are the eigenfunctions of the nominal beam, α_{ij}^0, α_{ij}^L are related to end inertia (B8 to B10 of Table 12.5.1), and β_{ij}^0, β_{ij}^L are related to springs at end springs (B5 to B10). Table 12.5.2 gives the coefficients α_{ij}^0, α_{ij}^L, β_{ij}^0, β_{ij}^L for different boundary conditions, where $\phi' = d\phi/dx$.

The kinetic energy and strain energy of the beam are functions of $q_j(t)$. The equations governing these generalized coordinates can be devised by Lagrange's equations. For this, consider the virtual work done by the external force $F(x, t)$ in Eq. (12.106) is

$$\delta W = \int_0^L F(x, t)\delta w(x, t)\,dx \tag{12.119}$$

TABLE 12.5.2 Coefficients α_{ij}^0, α_{ij}^L, β_{ij}^0, and β_{ij}^L

Boundary Conditions	α_{ij}^0	α_{ij}^L	β_{ij}^0	β_{ij}^L
B1 to B4	0	0	0	0
B5	0	0	$k_t\phi_i(0)\phi_j(0)$ $+k_r\phi_i'(0)\phi_j'(0)$	$k_t\phi_i(L)\phi_j(L)$ $+k_r\phi_i'(L)\phi_j'(L)$
B6	0	0	$k_r\phi_i'(0)\phi_j'(0)$	$k_r\phi_i'(L)\phi_j'(L)$
B7	0	0	$k_t\phi_i(0)\phi_j(0)$	$k_t\phi_i(L)\phi_j(L)$
B8	$m\phi_i(0)\phi_j(0)$ $+I\phi_i'(0)\phi_j'(0)$	$m\phi_i(L)\phi_j(L)$ $+I\phi_i'(L)\phi_j'(L)$	$k_t\phi_i(0)\phi_j(0)$ $+k_r\phi_i'(0)\phi_j'(0)$	$k_t\phi_i(L)\phi_j(L)$ $+k_r\phi_i'(L)\phi_j'(L)$
B9	$I_D\phi_i'(0)\phi_j'(0)$	$I_D\phi_i'(L)\phi_j'(L)$	$k_r\phi_i'(0)\phi_j'(0)$	$k_r\phi_i'(L)\phi_j'(L)$
B10	$m\phi_i(0)\phi_j(0)$	$m\phi_i(L)\phi_j(L)$	$k_t\phi_i(0)\phi_j(0)$	$k_t\phi_i(L)\phi_j(L)$

where δw is the virtual displacement of the beam. With the series (12.110), Eq. (12.119) is reduced to

$$\delta W = \sum_{j=1}^{N} p_j(t)\delta q_j(t) \tag{12.120}$$

where

$$p_j(t) = \int_0^L F(x,t)\phi_j(x)\,dx \tag{12.121}$$

is the generalized force corresponding to the generalized coordinate q_j. Substitute Eqs. (12.115) into Lagrange's equations for the beam problem

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} + \frac{\partial U}{\partial q_j} = p_j, \quad j = 1,2,\ldots \tag{12.122}$$

to yield the equations of motion of the discretized model

$$[M]\{\ddot{q}(t)\} + [K]\{q(t)\} = \{p(t)\} \tag{12.123}$$

where $\{p(t)\}$ is the vector of generalized forces

$$\{p(t)\} = (p_1(t) \quad p_2(t) \quad \cdots \quad p_N(t))^T \tag{12.124}$$

Equation (12.123) can also be derived though use of generalized Hamilton's principle.

Through a Rayleigh-Ritz discretization, the original partial differential equation (12.106) is reduced to a set of ordinary differential equations (12.123). Solution of these equations for the generalized coordinates leads to an approximation of the response of the nonuniform beam. In the subsequent sections, this discretized model will be used for dynamic analysis and feedback control of nonuniform beams.

12.5.3 Modes of Vibration

The eigenvalue problem of the nonuniform beam is derived by dropping $F(x,t)$ from Eq. (12.106) and by assuming $w(x,t) = u(x)e^{j\omega t}, j = \sqrt{-1}$, which yields

$$\rho(x)\omega^2 u(x) = \frac{\partial^2}{\partial x^2}\left[EI(x)\frac{\partial^2}{\partial x^2}u(x)\right], \quad 0 \le x \le L \tag{12.125}$$

Let the eigensolutions of Eq. (12.125) be $\omega_k, u_k(x), k = 1, 2, \ldots$, where ω_k is the kth eigenvalue (natural frequency) of the beam and $u_k(x)$ is the associate eigenfunction (mode shape). With the discretized model given in the previous section, the natural frequencies of the beam are the eigenvalues of the matrix eigenequation

$$\omega^2 [M]\{V\} = [K]\{V\} \tag{12.126}$$

where $[M]$ and $[K]$ are the mass and stiffness matrices in Eq. (12.123). The kth eigenvalues ω_k of Eq. (12.126) is the kth natural frequency of the beam; the associate eigenvector $\{V\}_k$ is related to the kth mode shape of the beam by

$$u_k(x) = [\Phi(x)]^T \{V\}_k \tag{12.127}$$

Here, $[\Phi(x)]$ is a vector of admissible functions in series (12.110), namely,

$$[\Phi(x)] = (\phi_1(x) \quad \phi_2(x) \quad \cdots \quad \phi_N(x))^T \tag{12.128}$$

The eigenvectors are usually scaled such that

$$\begin{aligned} \{V\}_j^T [M]\{V\}_k &= \delta_{jk} \\ \{V\}_j^T [K]\{V\}_k &= \omega_j \delta_{jk} \end{aligned} \tag{12.129}$$

where the delta function δ_{jk} is one for $j = k$ and zero for $j \neq k$.

The Toolbox of this chapter provides a MATLAB function `nonubeam` for computing the eigensolutions of a nonuniform Euler-Bernoulli beam; see Window 5.1.

Window 5.1. Function nonubeam

MATLAB Function: **nonubeam**

Purpose:
To set up a nonuniform Euler-Bernoulli beam and to compute its natural frequencies and mode shapes.

Synopsis:

```
nonubeam(rou_Spec, EI_Spec, L, BC_Spec)
nonubeam(rou_Spec, EI_Spec, L, BC_Spec, N)
[freq, v, msh] = nonubeam(rou_Spec, EI_Spec, L, BC_Spec, N)
```

Description:
`nonubeam(rou_Spec, EI_Spec, L, BC_Spec)` undertakes the following three tasks: (a) it inputs the linear density and bending stiffness of the beam by vectors `rou_Spec`

Window 5.1. Function nonubeam (*Continued*)

and EI_Spec, respectively, and specifies beam length L; (b) it assigns the boundary conditions by a vector BC_Spec; and (c) it computes and displays the eigenvalues (natural frequencies) and eigenfunctions (mode shapes) of the beam. The construction of vectors rou_Spec and EI_Spec is given in Table 12.5.3; the construction of BC_Spec is given in Table 12.5.4.

nonubeam(rou_Spec, EI_Spec, L, BC_Spec, N) specifies the number N of terms in the Rayleigh-Ritz approximation described by Eq. (12.110). By default, N = 20.

[freq, v, msh] = nonubeam(rou_Spec, EI_Spec, L, BC_Spec, N) returns the natural frequencies of the beam in a vector freq; the eigenvectors of the discretized model of the beam in a matrix v; and the coefficients of the normalized eigenfunctions of the nominal beam by a matrix msh. The *k*th column of matrix v is the *k*th eigenvector of the eigenequation (12.126). The contents of msh are the same as those described in Window 2.2. With v and msh, the eigenfunctions of the nonuniform beam are given by Eq. (12.127).

Note. For the information on the parameters, boundary conditions, and eigensolutions of a nonuniform beam that are assigned/obtained by function nonubeam, type systinfo2 in the MATLAB command window.

Besides nonubeam, the Toobox also offers function plotmsh2 for plotting mode shapes and function animode2 for animating modes of vibration of nonuniform beams; see Windows 5.2 and 5.3.

Window 5.2. Function plotmsh2

MATLAB Function: **plotmsh2**

Purpose:
To plot a mode shape of a nonuniform beam.

Synopsis:

```
plotmsh2(n)
plotmsh2(n, npts)
[y, x, freq, nodes] = plotmsh2(n, npts)
```

Description:
plotmsh2(n) plots the *n*th mode shape (modal displacement) of a nonuniform beam over its spatial region [0, *L*].

plotmsh2(n, npts) selects the number npts of equally-spaced points in the region [0, *L*] for plotting the mode shape. By default, npts = 201.

[y, x, w, nodes] = plotmsh2(n, npts) returns the spatial distribution of the *n*th mode shape of the beam in a vector y, spatial points of computation in a vector x, the natural frequency of the mode in w, and the nodal points of the mode shape in a vector nodes. With x and y, the mode shape can be plotted by the MATLAB function plot(x, y).

TABLE 12.5.3 Specification of Linear Density and Bending Stiffness for Nonuniform Beams

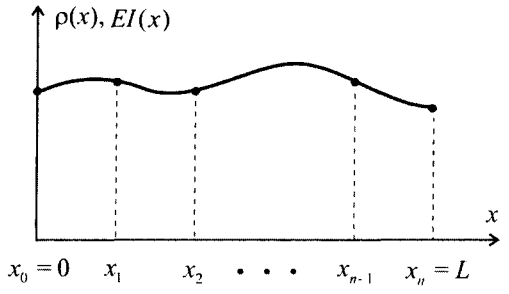
Beam Parameters	*Specification*
Type 0 (uniform) $\rho(x) = \rho_0 = \text{constant}$ $EI(x) = EI_0 = \text{constant}$	rou_Spec = ρ_0 EI_Spec = EI_0
Type 1 (linear or tapered) $\rho(x) = \rho_0 (1 - x/L) + \rho_L x/L$ $EI(x) = EI_0 (1 - x/L) + EI_L x/L$	rou_Spec = [1 ρ_0 ρ_L] EI_Spec = [1 EI_0 EI_L]
Type 2 (polynomial) $\rho(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$ $EI(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$	rou_Spec = [2 c_0 c_1 c_2 $\cdots$ c_n] EI_Spec = [2 c_0 c_1 c_2 $\cdots$ c_n]
Type 3 (sinusoidal) $\rho(x) = a + b\sin(cx + \theta_0)$ $EI(x) = a + b\sin(cx + \theta_0)$	rou_Spec = [3 a b c θ_0] EI_Spec = [3 a b c θ_0] θ_0 in degrees
Type 4 (irrational) $\rho(x) = a[c_0 + c_1 x + \cdots + c_n x^n]^b$ $EI(x) = a[c_0 + c_1 x + \cdots + c_n x^n]^b$ b – an arbitrary number	EI_Spec = [4 a b c_0 c_1 $\cdots$ c_n] EI_Spec = [4 a b c_0 c_1 $\cdots$ c_n]
Type 5 (exponential) $\rho(x) = a + be^{cx}$ $EI(x) = a + be^{cx}$	rou_Spec = [5 a b c] EI_Spec = [5 a b c]
Type 6 (user-provided)  	rou_Spec = [6 $\rho(x_0)$ $\rho(x_1)$ $\cdots$ $\rho(x_n)$] EI_Spec = [6 $EI(x_0)$ $EI(x_1)$ $\cdots$ $EI(x_n)$] Here $x_k = k\Delta, k = 0, 1, \ldots, n, \Delta = L/n$

TABLE 12.5.4 Specification of Boundary Conditions for Nonuniform Beams

`BC_Spec = [BC_Left  BC_Right]`

(B1) Pinned or hinged end

`BC_Left = 1;  BC_Right = 1`

(B2) Clamped or fixed end

`BC_Left = 2;  BC_Right = 2`

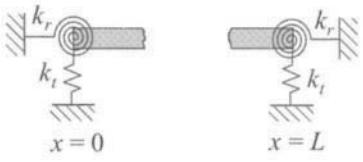
(B3) Free end

`BC_Left = 3;  BC_Right = 3`

(B4) Sliding or guided end

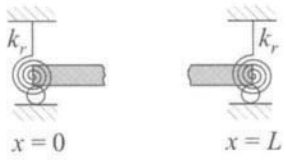
`BC_Left = 4;  BC_Right = 4`

(B5) Elastic-support



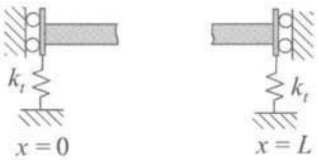
`BC_Left = [5 k_r k_t];  BC_Right = [5 k_r k_t]`

(B6) Hinged-elastic end



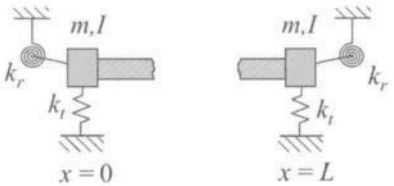
`BC_Left = [6 k_r];  BC_Right = [6 k_r]`

(B7) Sliding-elastic end



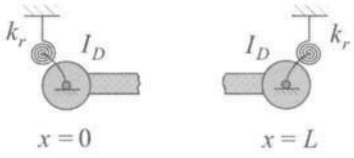
`BC_Left = [7 k_t];`
`BC_Right = [7 k_t]`

(B8) End mass



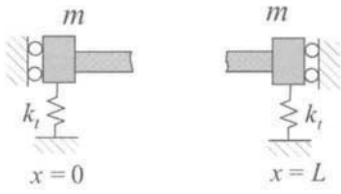
`BC_Left = [8 m k_t I k_r]`
`BC_Right = [8 m k_t I k_r]`

(B9) Hinged disk



`BC_Left = [9 I_D k_r];`
`BC_Right = [9 I_D k_r]`

(B10) Sliding mass



`BC_Left = [10 m k_t];`
`BC_Right = [10 m k_t]`

Window 5.3. Function animode2

MATLAB Function: **animode2**

Purpose:
To animate a mode of vibration of a nonuniform beam.

Synopsis:

```
animode2(k)
F = animode2(k, tf, n_frames)
```

Description:
`animode2(k)` animates the *k*th mode of vibration of a nonuniform beam.

`F = animode2(k, tf, n_frames)` returns a set of frames of beam vibration in the *k*th mode in a matrix `F`, which can be played later on by the MATLAB function `movie(F)`. Here `tf` is total animation time and `n_frames` is the number of frames in the animation. By default, `tf` = 8*T, with T being the period of the mode, and `n_frames` = 97.

EXAMPLE 5.1

In Fig. 12.5.2, a nonuniform beam is hinged at its left end with a rigid disk mounted on it, and has an end mass at its right end. The beam with B9-B8 boundaries can be viewed as a model of flexible manipulators or disk drive read/write servo systems. Let the parameters of the beam be

$$\rho(x) = 0.5(1 - x/L) + 0.25x/L$$

$$EI(x) = 10 \cdot \sqrt[4]{1 - 0.2x^2}$$

$$L = 2, \quad I_D = 5, \quad m = 0.5, \quad I = 0.2.$$

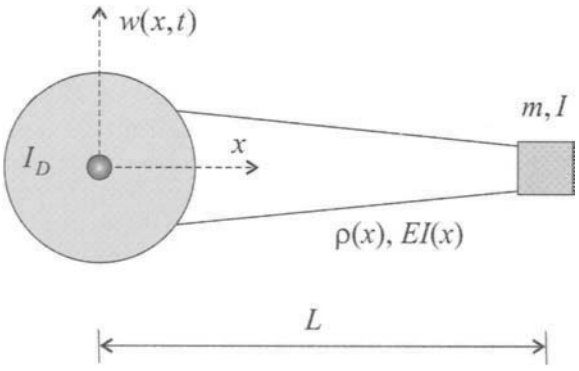


FIGURE 12.5.2

```
>> rou = [1 0.5 0.25];
>> EI = [4 10 0.25 1 0 -0.2];
>> BC = [9 5 0 8 0.5 0 0.2 0];
>> nonubeam(rou, EI, 2, BC)
```

yields the first eight natural frequencies of the beam as follows

```
Wn, natural frequencies in rad/s
fn = wn/(2*pi), natural frequencies in Hertz

     wn        fn
      0         0
 2.7865    0.4435
 9.3495    1.4880
32.9877    5.2502
84.6023   13.4649
162.7434  25.9014
268.8912  42.7954
399.6462  63.6057
```

and plots the first four mode shapes in Fig. 12.5.3. Function `nonubeam` also plots the spatial distribution of $\rho(x)$ and $EI(x)$ (not shown herein). Moreover, through use of function `plotmsh2`:

```
>> [y1,x]=plotmsh2(1);
>> y2 = plotmsh2(3); y3 = plotmsh2(5);
>> plot(x, y1, x, y2, ':', x, y3,'-.')
>> legend('1st mode','3rd mode', '5th mode')
>> grid; xlabel('x')
```

the 1st, 3rd, and 5th mode shapes are plotted in Fig. 12.5.4.

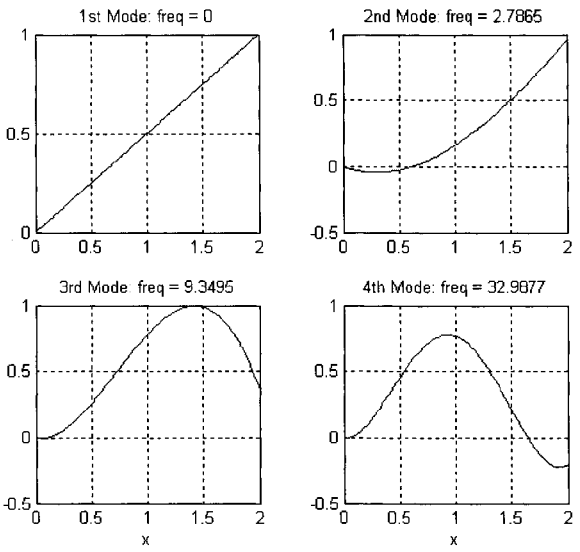


FIGURE 12.5.3

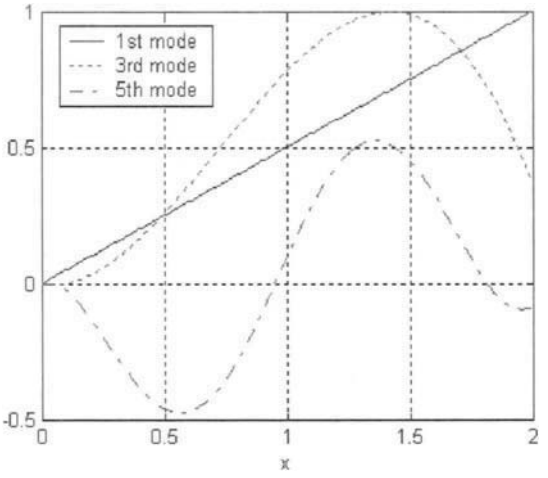


FIGURE 12.5.4

12.5.4 Time Response

Assume that a nonuniform beam is subject to an external force

$$F(x,t) = D(x)f(t) \tag{12.130}$$

where $D(x)$ describes the spatial distribution of the force. By the discretization process described in Section 12.5.2, Eqs. (12.106) to (12.108) are reduced to the following ordinary differential equations

$$[M]\{\ddot{q}(t)\} + [K]\{q(t)\} = \{B\}f(t) \tag{12.131}$$

subject to the initial conditions

$$\{q(0)\} = \{q_0\}, \quad \{\dot{q}(0)\} = \{r_0\} \tag{12.132}$$

where the mass and stiffness matrices $[M]$, $[K]$, the vector $\{q(t)\}$ of generalized coordinates have been given in Section 12.5.2, the force distribution vector

$$\{B\} = (b_1 \quad b_2 \quad \cdots \quad b_N)^T, \quad \text{with } b_k = \int_0^L D(x)\phi_k(x)\,dx \tag{12.133}$$

and the initial disturbance vectors

$$\{q_0\} = \bar{\rho}\int_0^L a_0(x)\,[\Phi(x)]\,dx, \quad \{r_0\} = \bar{\rho}\int_0^L b_0(x)\,[\Phi(x)]\,dx \tag{12.134}$$

Here, $a_0(x)$ and $b_0(x)$ are the distributions of initial displacement and velocity of the nonuniform beam, $\bar{\rho}$ and $\phi_k(x)$ are the linear density and eigenfunctions of the nominal beam that are used in the Rayleigh-Ritz discretization, and $[\Phi(x)]$ is given in Eq. (12.128).

The elements of vector $\{B\}$ are derived for the following three cases:

Case 1. A pointwise transverse force applied at $x = x_f$, $F(x,t) = \delta(x - x_f) f(t)$.

$$b_k = \int_0^L \delta(x - x_f)\phi_k(x)\, dx = \phi_k(x_f) \tag{12.135a}$$

Case 2. A force uniformly distributed in region $x_1 \le x \le x_2$,
$F(x,t) = (h(x - x_1) - h(x - x_2))f(t)$, where $h(x)$ is the unite step function.

$$b_k = \int_{x_1}^{x_2} \phi_k(x)\, dx \tag{12.135b}$$

Case 3. A pointwise torque applied at $x = x_\tau$, $F(x,t) = -\frac{d}{dx}\delta(x - x_f) f(t)$.

$$b_k = -\int_0^L \frac{d\delta(x - x_\tau)}{dx}\phi_k(x)\, dx = \frac{d\phi_k(x_\tau)}{dx} \tag{12.135c}$$

See Fig. 12.3.5 for the illustration of the above three types of force distributions.

Solution of Eqs. (12.131) and substitution of the result into the series (12.110) gives the time response of the beam. This Toolbox offers several functions for prediction of time response of nonuniform beams. The solution procedure takes the following steps:

Step 1. Given a nonuniform beam, determine the eigenfunctions $\phi_k(x)$ of the corresponding nominal beam defined in Section 12.5.2;
Step 2. With $\phi_k(x)$, compute quantities $[M]$, $[K]$, $\{B\}$, $\{q_0\}$, and $\{r_0\}$ in Eqs. (12.131) and (12.132) by function `getmkb0`;
Step 3. Solve the ordinary differential equations (12.132) for the generalized coordinates $q_k(t)$; and
Step 4. With the determined $q_k(t)$, plot and animate the response of the nonuniform beam by functions `respnubeam` and `nubmovie`.

The task in Step 1 is automatically undertaken by function `nonubeam`. Step 3 can be carried out through use of the MATLAB toolbox of Chapter 11. The utility of functions `getmkb0`, `respnubeam`, and `nubmovie` is described in Windows 5.4 to 5.6.

Window 5.4. Function `getmkb0`

MATLAB Function: **`getmkb0`**

Purpose:
To get $[M]$, $[K]$, $\{B\}$, $\{q_0\}$, and $\{r_0\}$ in Eqs. (12.131) and (12.132) for a nonuniform beam.

Synopsis:

```
[M, K, B, qr] = getmkb0(FD_Spec, IC)
getmkb0(FD_Spec, IC)
```

Window 5.4. Function getmkb0 (*Continued*)

Description:
`[M, K, B, qr] = getmkb0(FD_Spec, IC)` obtains the mass matrix $[M]$, stiffness matrix $[K]$, and force distribution vector $\{B\}$ of Eq. (12.131) in `M`, `K`, and `B`, respectively, and the initial disturbances vectors $\{q_0\}$, $\{r_0\}$ of Eq. (12.132) in a two-column matrix `qr`, with the first column `qr(:,1)` storing the elements of $\{q_0\}$ and the second column `qr(:,2)` storing the elements of $\{r_0\}$. Here, `FD_Spec` is a vector specifying one of the three force distributions given in Eqs. (12.135) as follows:

Case (a) A pointwise transverse force at $x = x_f$: `FD_Spec = [0` x_f`]`

Case (b) A force uniformly distributed in region $x_1 \leq x \leq x_2$: `FD_Spec = [1` x_1 x_2`]`

Case (c) A pointwise torque at $x = x_\tau$: `FD_Spec = [2` x_τ`]`

The `IC` is a 2-by-*np* matrix specifying the initial disturbances of the beam at *np* equally spaced points along the beam length:

$$\texttt{IC} = \begin{bmatrix} a_0(x_1) & a_0(x_2) & \cdots & a_0(x_{np}) \\ b_0(x_1) & b_0(x_2) & \cdots & b_0(x_{np}) \end{bmatrix}; \quad x_k = \frac{(k-1)L}{np-1}, \quad k = 1, 2, \ldots, np$$

where $a_0(x)$ and $b_0(x)$ are the initial displacement and velocity given in Eq. (12.108), and L is the beam length. For accurate results, number *np* should be large enough, say $np > 500$. If the beam has zero initial disturbances, simply assign `IC = 0`. In addition, `[M, K, B] = getmkb0(FD_Spec)` obtains the above-mentioned $[M]$, $[K]$, and $\{B\}$ for a nonuniform beam under zero initial disturbances.

`getmkb0(FD_Spec, IC)` assigns the above-mentioned quantities as global variables in the MATLAB workspace, which can be accessed by other functions from the Toolbox.

Note:
Function `nonubeam` must be executed before `getmkb0` can be used.

Window 5.5. Function respnubeam

MATLAB Function: **respnubeam**

Purpose:
To plot the time response of a nonuniform beam.

Synopsis:

```
respnubeam(xr, q, t)
y = respnubeam(xr, q, t)
```

Window 5.5. Function respnubeam (*Continued*)

Description:
`respnubeam(xr, q, t)` plots the time response (displacement, slope, bending moment, and shear force) of the beam at point `xr`, where `q` is a matrix containing the time history data of the generalized coordinate vector $\{q(t)\}$ in Eq. (12.131), and `t` is a vector of computation times. The kth row of `q`, `q(k,:)`, is the time history of the kth generalized coordinate $q_k(t)$. The `q` and `t` can be obtained through use of the MATLAB functions, such as `freeRK` and `forcedRK` of Chapter 11 (Vibration and Control of Multiple-Degree-of-Freedom Systems).

`y = respnubeam(xr, q, t)` returns the time response of the beam at `xr` in a four-row matrix, with the first row `y(1,:)` being the displacement $w(x_r, t)$, second row `y(2,:)` the rotation (slope) $\theta(x_r, t)$, the third row `y(3,:)` the bending moment $M(x_r, t)$, and the last row `y(4,:)` the shear force $Q(x_r, t)$.

Note:
Function `nonubeam` must be executed before `respnubeam` can be used.

Window 5.6. Function nubmovie

MATLAB Function: **nubmovie**

Purpose:
To animate the time response of a nonuniform beam in its the spatial region.

Synopsis:

```
nubmovie(q, t)
nubmovie(q, t, IS_control, n_frames, frame_bounds, npts)
[F, y, t_frames] = nubmovie(q, t, IS_control, n_frames,
frame_bounds, npts)
```

Description:
`nubmovie(q, t)` plays animated vibration of a nonuniform beam, where `q` is a matrix containing the time history data of $\{q(t)\}$ in Eq. (12.131), and `t` is a vector of computation times. The `q, t` can be obtained by the MATLAB functions of Chapter 11, such as `freeRK` and `forcedRK`.

`nubmovie(q, t, IS_control, n_frames, frame_bounds, npts)` plays animated vibration of a beam in a specified manner, where `IS_control` is an animation control parameter with the following options:

`IS_control = 0` play frames continuously
`IS_control = 1` play frames one by one

Window 5.6. Function nubmovie (*Continued*)

`n_frames` is the total number of frames that are to be played in the time spanned by the vector `t`; `frame_bounds = [xmin xmax ymin ymax]` sets up the spatial bounds of frames; and `npts` is the number of spatial points along the beam length selected for animation. By default, `IS_control = 0`, `n_frames = 101`, `frame_bounds` is automatically set up according to the given data `q`, and `npts = 201`. The `IS_control`, `n_frames`, `frame_bounds`, and `npts` can be replaced by [ ] as a placeholder for their default setup.

`[F, t_frames, y] = nubmovie(q, t, IS_control, n_frames, frame_bounds, npts)` returns a set of frames of beam vibration in a matrix `F`, which later on can be played back by the MATLAB function `movie(F)`, the times of animation in a vector `t_frames`, and beam displacement profile at the animation times in a matrix `y`, with the kth row `y(k,:)` consisting of the spatial distribution of the beam displacement at time `t_frames(k)`.

EXAMPLE 5.2

In Fig. 12.5.5, a nonuniform beam with elastically hinged-hinged boundaries is subject to a constant moment $M_0(t) = 2.5$ at its left end. Let the beam parameters be

$$\rho(x) = 0.5(1 - x/L) + 0.25x/L, \quad EI(x) = 10 \cdot \sqrt[4]{1 - 0.2x^2}, \quad L = 2, \quad k_r = 10$$

The beam is initially at rest. Assume that each mode of vibration of the beam has 5% modal damping. Let the beam be approximated by the series (12.110) with N = 8. The beam response at $x = 1.2$ is computed in the following three steps:

(i) Set up the beam and compute matrices $[M]$, $[K]$, and vector $\{B\}$ in Eq. (12.131):

```
>> rou = [1 0.5 0.25];
>> EI = [4 10 0.25 1 0 -0.2];
>> nonubeam(rou, EI, 2, [6 10 1], 8)
>> [M, K, Bf] = getmkb0([2 0]);
```

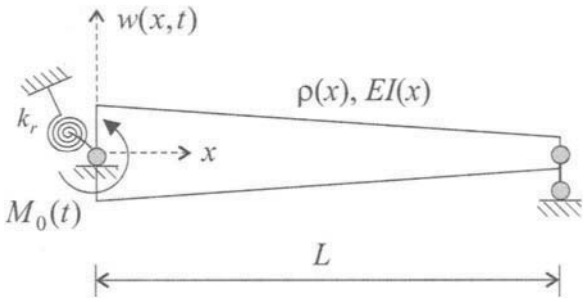


FIGURE 12.5.5

(ii) Compute the generalized coordinates $\{q(t)\}$ by function `forcedRK` of Chapter 11 (Vibration and Control of Multiple-Degree-of-Freedom Systems):

```
dmp = [2 0.05]; % modal damping of 5% for each mode
setmdof(M,K,dmp)
[q, t] = forcedRK([1 2.5], Bf, 2, 4001);
```

To use functions `setmdof` and `forcedRK`, set the current directory of the MATLAB command window to where the Toolbox of Chapter 11 is located on the computer.

(iii) Set the current directory of MATLAB command window back to the Toolbox of Chapter 12 (this chapter) and plot the response of the beam as follows

```
>> nonubeam(rou, EI, 2, [6 10 1], 8)
>> respnubeam(1.2, q, t)
```

which yields Fig. 12.5.6.

Furthermore, function `nubmovie` is used to animate the vibration of the beam in the time period $0 \leq t \leq 0.1$. This is done by

```
% Use of the Toolbox of Chapter 11
setmdof(M,K, [2 0.05])
[q, t] = forcedRK([1 2.5], Bf, 0.1, 4001);

% Use of the Toolbox of Chapter 12
nonubeam(rou, EI, 2, [6 10 1], 8)
nubmovie(q, t, 1, 6)
```

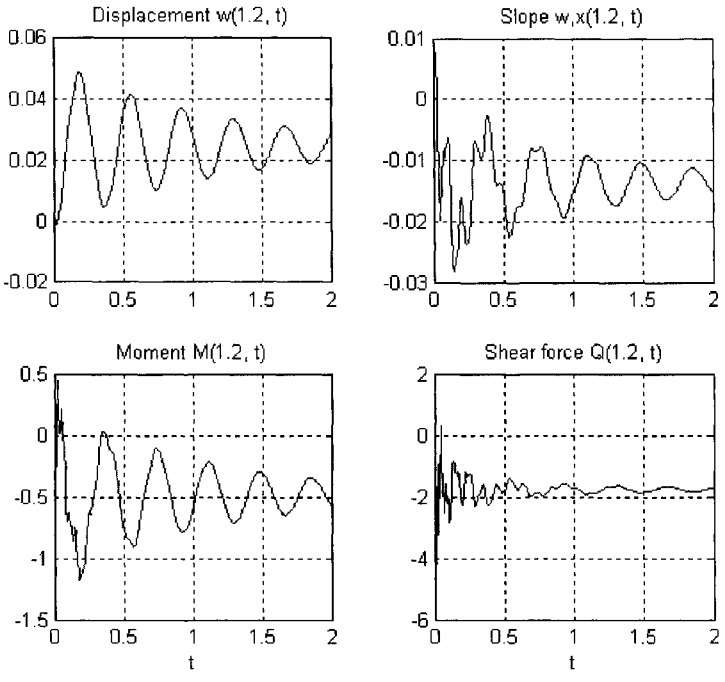


FIGURE 12.5.6

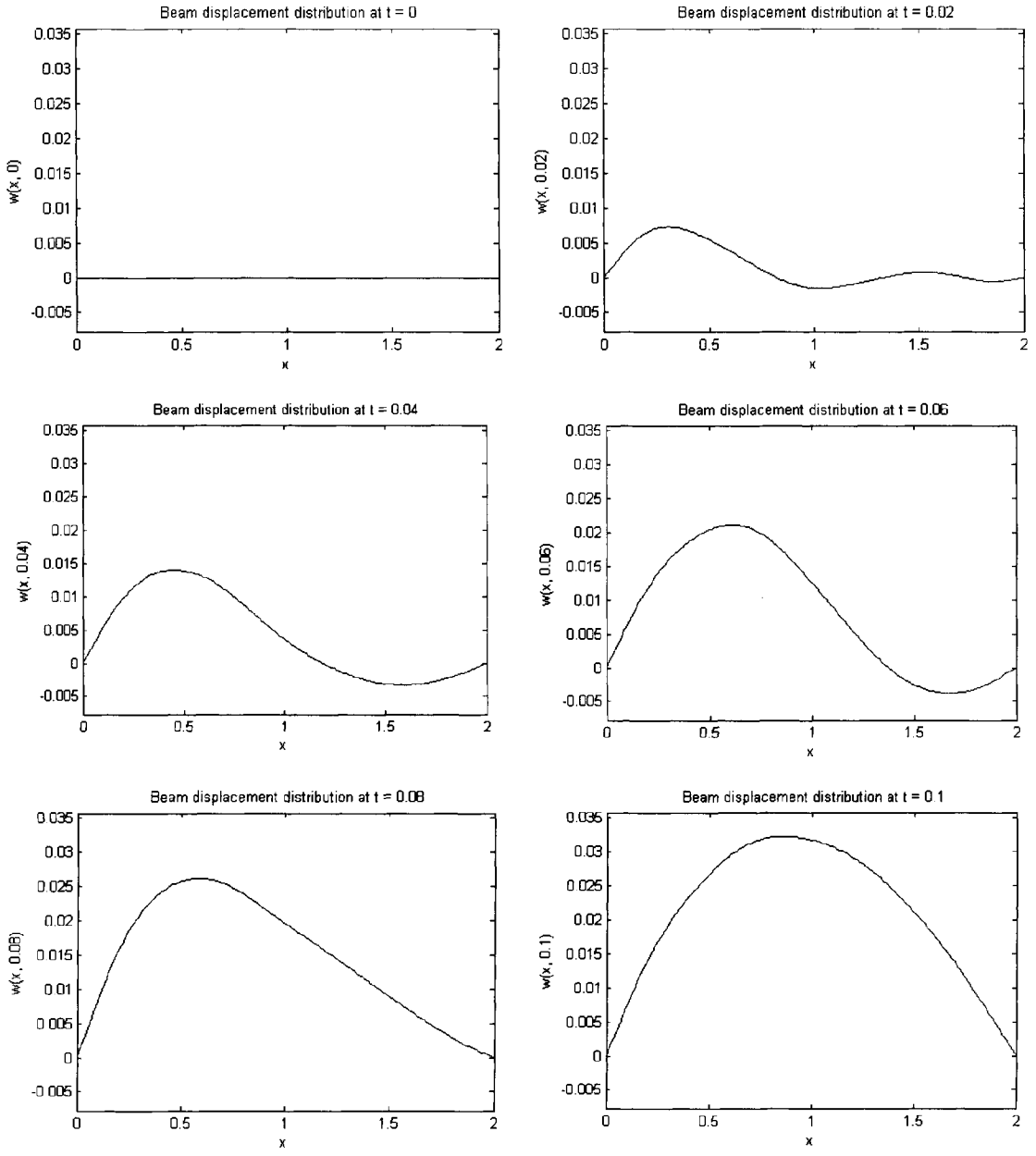

Beam displacement distribution at t = 0
w(x, 0)
Beam displacement distribution at t = 0.02
w(x, 0.02)
Beam displacement distribution at t = 0.04
w(x, 0.04)
Beam displacement distribution at t = 0.06
w(x, 0.06)
Beam displacement distribution at t = 0.08
w(x, 0.08)
Beam displacement distribution at t = 0.1
w(x, 0.1)
x

FIGURE 12.5.7 Animated beam response.

which yields six frames of the beam displacement profiles in Fig. 12.5.7.

Additionally, the beam displacement profile at the times of the six frames in Fig, 12.5.7 can be plotted as follows

```
>> [F, t, y] = nubmovie(q, t, 1, 6);
>> npts = length(y(1,:));
>> xx = linspace(0, 2, npts);
```

```
>> plot(xx,y)
>> xlabel('x'), ylabel('w(x,t)')
>> title('w(x,t) at t = 0, 0.02, 0.04, 0.06, 0.08, 0.1')
```

which yields Fig. 12.5.8.

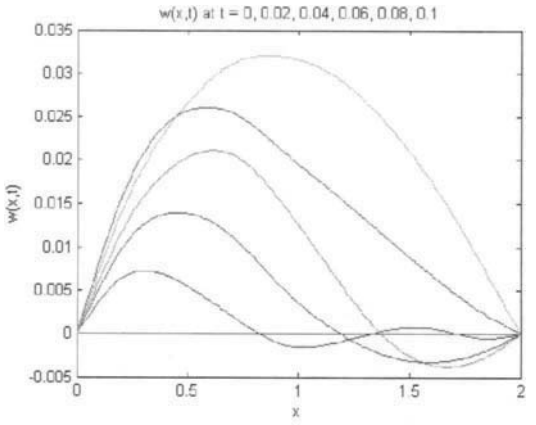


FIGURE 12.5.8

12.5.5 Control System Formulation

Transfer Function Formulation In Fig. 12.5.9, a nonuniform beam is controlled via a pointwise sensor located at x_s and a pointwise actuator located at x_a, where $u(t)$ is the control force applied to the beam by the actuator, and $y(t)$ is the sensor output. The equation governing the controlled beam is

$$\rho(x)\frac{\partial^2}{\partial t^2}w(x,t) + \frac{\partial^2}{\partial x^2}\left[EI(x)\frac{\partial^2}{\partial x^2}w(x,t)\right] = D_a(x)\,u(t), \quad 0 \le x \le L \tag{12.136}$$

where $D_a(x)$ is the distribution of the control force $u(t)$. The sensor output is expressed as

$$y(t) = D_s\,[w(x,t)]_{x=x_s} \tag{12.137}$$

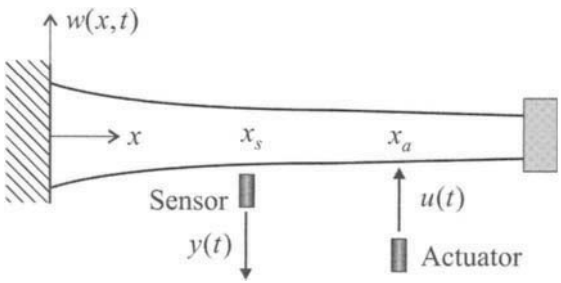


FIGURE 12.5.9 A nonuniform beam with a pair of sensor and actuator.

TABLE 12.5.5 Actuator and Control Force

Actuator Type	*Coefficient* a_k	`Actuator_Spec`
(A1) Transverse force $D_a(x)u(t) = \delta(x - x_a)\,u(t)$	$[\Phi(x_a)]^T \{V\}_k$	[1 x_a]
(A2) Moment $D_a(x)u(t) = -\dfrac{d}{dx}\delta(x - x_a)\,u(t)$	$\dfrac{d}{dx}[\Phi(x_a)]^T \{V\}_k$	[2 x_a]

TABLE 12.5.6 Sensor and Output Signal

Sensor Type	*Coefficient* b_k	`Sensor_Spec`
(S1) Transverse displacement $y(t) = w(x_s, t)$	$[\Phi(x_s)]^T \{V\}_k$	[1 x_s]
(S2) Slope or rotation $y(t) = \dfrac{\partial}{\partial x} w(x_s, t)$	$\dfrac{d}{dx}[\Phi(x_s)]^T \{V\}_k$	[2 x_s]
(S3) Strain $y(t) = \dfrac{\partial^2}{\partial x^2} w(x_s, t)$	$\dfrac{d^2}{dx^2}[\Phi(x_s)]^T \{V\}_k$	[3 x_s]

where D_s is a spatial differential operator. Tables 12.5.5 and 12.5.6 list several commonly used types of sensors and actuators.

Like in the case of a uniform beam, the transfer function of the nonuniform beam can be expressed in a modal series. Following Section 12.4.1, the transfer function from the control force to the sensor output is obtained as

$$G_o(s) = \frac{Y(s)}{U(s)} = \sum_{k=1}^{N} \frac{a_k b_k}{s^2 + \omega_k^2} \tag{12.138}$$

where the coefficients a_k and b_k are related to the sensor and actuator locations, and are given in Tables 12.5.5 and 12.5.6. In the tables, $[\Phi(x)]$ is a vector of eigenfunctions defined in Eq. (12.128), and $\{V\}_k$ is the kth eigenvector of Eq. (12.126). In engineering analysis, modal damping is often introduced. In that case, the transfer function can be written as

$$G_o(s) = \frac{Y(s)}{U(s)} = \sum_{k=1}^{N} \frac{a_k b_k}{s^2 + c_k + \omega_k^2} \tag{12.139}$$

where $c_k = 2\xi_k \omega_k$, with ξ_k being the damping ratio of the kth mode.

The Toolbox of this chapter offers a function `tfbeam2` for obtaining the transfer function given in Eqs. (12.138) and (12.139); see Window 5.7.

Window 5.7. Function tfbeam2

MATLAB Function: **tfbeam2**

Purpose:
To create the transfer function $G_o(s)$ of a nonuniform beam given in Eq. (12.139).

Synopsis:

```
tfbeam2(Sensor_Spec, Actuator_Spec)
tfbeam2(Sensor_Spec, Actuator_Spec, Dmp_Spec, n_mode)
[num, den, p, z, abc] = tfbeam(Sensor_Spec, Actuator_Spec,
Dmp_Spec, n_mode)
```

Description:
`tfbeam2(Sensor_Spec, Actuator_Spec)` displays the transfer function $G_o(s)$ and shows its poles and zeros in the complex plane, where `Sensor_Spec` and `Actuator_Spec` specify the sensor/actuator type and location by Tables 12.5.5 and 12.5.6.

`tfbeam2(Sensor_Spec, Actuator_Spec, Dmp_Spec, n_mode)` displays the transfer function with specified parameters. Here, `Dmp_Spec` is a vector that specifies modal damping in the beam with the following options:

(i) No damping, `Dmp_Spec = 0 or [ ]`.
(ii) `Dmp_Spec = [1 zeta_1, zeta_2 ... zeta_m]`, where `zeta_k` is the modal damping ratio of the kth mode. All damping ratios fall between 0 and 1. If $m < n$, where n is the total number of modes considered, the $(m+1)$th ,..., nth modes have the same damping ratio as `zeta_m`. For instance, `Dmp_Spec` = [2 0.1 0.05] specifies a damping ratio of 0.1 for the first mode, and a damping ratio of 0.05 for the reset modes.

The parameter `n_mode` specifies the number of modes of the beam selected for construction of the transfer function. By default, `Dmp_Spec = 0` and `n_mode = min(4, N)` with `N` being the number of the terms in the series (12.110) that is set up by function `nonubeam`.

`[num,den,p,z,abc] = tfbeam(Sensor_Spec, Actuator_Spec, Dmp_Spec, n_mode)` returns the coefficients of the numerator and denominator of $G_o(s)$ in vectors `num` and `den`, respectively, the poles and zeros of $G_o(s)$ in vectors `p` and `z`, respectively, and the coefficients in the modal series (12.139) in a four-row matrix `abc`, with the first row storing the coefficients a_k, the second row b_k, the third row c_k, and the fourth row ω_k. With vectors `num` and `den`, $G_o(s)$ can be constructed by `tf(num, den)`, where `tf` is a function from the MATLAB Control System Toolbox.

EXAMPLE 5.3

Consider the same beam as given in Example 5.2. Let a slope sensor be placed at the right end ($x = L$) of the beam. Let a moment actuator be placed at the left end ($x = 0$). Assume that the first three modes of vibration of the beam are considered in transfer function formulation. Each mode has 5% modal damping. The commands

```
>> rou = [1 0.5 0.25];
>> EI = [4 10 0.25 1 0 -0.2];
```

```
>> nonubeam(rou, EI, 2, [6 10 1])
>> [num, den] = tfbeam2([2 2], [2 0], 0.05, 3);
```

yield the transfer function of the beam from the moment of the actuator to the sensor output as follows

```
Transfer Function in Modal Series
---------------------------------
Number of modes, N = 3
G(s) = Y(s)/U(s) = Term 1 + Term 2 + Term 3
where
                  d1
Term 1 = ------------------, with d1 = -5.6293, c1 = 1.4247,
         s^2 + c1*s + w1^2        w1 = 14.2474

                  d2
Term 2 = ------------------ , with d2 = 24.1714, c2 = 5.1288,
         s^2 + c2*s + w2^2         w2 = 51.2877

                  d3
Term 3 = ------------------, with d3 = -57.2154, c3 = 11.2703,
         s^2 + c3*s + w3^2        w3 = 112.7033
```

and plot the poles and zeros of the transfer function in Fig. 12.5.10. In addition,

```
>> tf(num,den)
```

gives the transfer function in the standard rational form:

```
Transfer function:
-38.67 s^4 - 160.4 s^3 + 6.315e004 s^2 - 3.149e005 s - 1.563e008
-----------------------------------------------------------------
s^6 + 17.82 s^5 + 1.562e004 s^4 + 1.2e005 s^3 + 3.667e007 s^2 +
6.684e007 s + 6.782e009
```

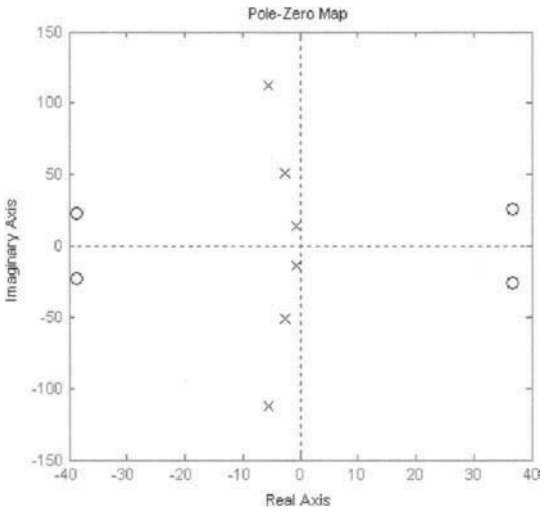


FIGURE 12.5.10

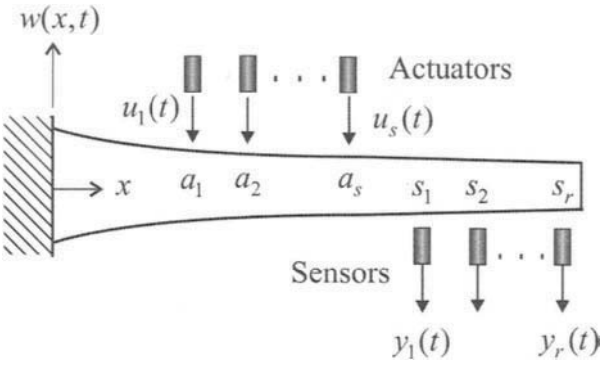


FIGURE 12.5.11 A nonuniform beam with multiple sensors and actuators.

State Representation In Fig. 12.5.11, a nonuniform beam is under feedback control with r pointwise sensors located at $s_1, s_2, \ldots, s_r$, and s pointwise actuators located at $a_1, a_2, \ldots, a_s$. The open-loop system (beam plus sensors and actuators) is described by

$$\rho(x)\frac{\partial^2}{\partial t^2}w(x,t) + \frac{\partial^2}{\partial x^2}\left[EI(x)\frac{\partial^2}{\partial x^2}w(x,t)\right] = \sum_{j=1}^{s} D_j(x)\,u_j(t), \quad 0 \le x \le L \tag{12.140}$$

$$y_k(t) = E_k\,[w(x,t)]_{x=a_k}, \quad k = 1,2,\ldots,r \tag{12.141}$$

where $u_j(t)$ is the control force by the jth actuator, $y_k(t)$ is the output signal of the kth sensor, $D_j(x)$ are the force distribution functions, and E_k are spatial differential operators. With the discretization process in Section 12.5.2, Eqs. (12.140) and (12.141) are reduced to

$$[M]\,\{\ddot{q}(t)\} + [K]\,\{q(t)\} = [B_a]\,\{u(t)\} \tag{12.142}$$

$$\{y(t)\} = [C_s]\,\{q(t)\} \tag{12.143}$$

where $\{q(t)\}$ is the vector of generalized coordinates, $[M]$ and $[K]$ are mass and stiffness matrices given in Eq. (12.123), and $\{u(t)\}$ and $\{y(t)\}$ are the vector of control forces and vector of sensor output signals, i.e.,

$$\{u(t)\} = \begin{pmatrix} u_1(s) \\ u_2(s) \\ \vdots \\ u_r(s) \end{pmatrix}, \quad \{y(t)\} = \begin{pmatrix} y_1(s) \\ y_2(s) \\ \vdots \\ y_s(s) \end{pmatrix} \tag{12.144}$$

The sensors and actuators are described in Tables 12.5.5 and 12.5.6. The matrices $[B_a]$ and $[C_s]$ are related to the type and location of each sensor and actuator by

$$[B_a] = [\{\alpha_1\} \quad \{\alpha_2\} \quad \cdots \quad \{\alpha_r\}], \quad [C_s] = [\{\beta_1\} \quad \{\beta_2\} \quad \cdots \quad \{\beta_s\}]^T \tag{12.145}$$

with

$$\{\alpha_j\} = \begin{cases} [\Phi(a_j)] & \text{for force actuator (A1)} \\ \dfrac{d}{dx}[\Phi(a_j)] & \text{for moment actuator (A2)} \end{cases} \tag{12.146a}$$

$$\{\beta_k\} = \begin{cases} [\Phi(s_k)] & \text{for displacement actuator (S1)} \\ \dfrac{d}{dx}[\Phi(s_k)] & \text{for slope sensor (S2)} \\ \dfrac{d^2}{dx^2}[\Phi(s_k)] & \text{for strain sensor (S3)} \end{cases} \tag{12.146b}$$

By defining the state vector

$$\{z(t)\} = \begin{Bmatrix} \{q(t)\} \\ \{\dot{q}(t)\} \end{Bmatrix} \tag{12.147}$$

Eqs. (12.142) and (12.143) are converted to the state equation

$$\{\dot{z}(t)\} = [A]\{z(t)\} + [B]\{u(t)\} \tag{12.148}$$

and the output equation

$$\{y(t)\} = [C]\{z(t)\} \tag{12.149}$$

where

$$[A] = \begin{bmatrix} [0] & [I] \\ -[M]^{-1}[K] & [0] \end{bmatrix}, \quad [B] = \begin{bmatrix} [0] \\ [B_c] \end{bmatrix}, \quad [C] = \begin{bmatrix} [C_s] & [0] \end{bmatrix} \tag{12.150}$$

with [0] and $[I]$ being zero matrix and identity matrix of appropriate order.

This Toolbox offers a function `ssbeam2` for obtaining the state equation (12.148) and the sensor output equation (12.149) for a nonuniform beam; see Window 5.8.

Window 5.8. Function ssbeam2

MATLAB Function: **ssbeam2**

Purpose:
To create a state-space model for a nonuniform beam under r sensors and s actuators.

Synopsis:

```
[A, B, C] = ssbeam2(Sensor_Spec, Actuator_Spec, Dmp_Spec)
```

Description:
`[A, B, C] = ssbeam2(Sensor_Spec, Actuator_Spec, Dmp_Spec)` returns matrices $[A]$, $[B]$, and $[C]$ of the state representation described by Eqs. (12.148) and (12.149). Here, `Sensor_Spec` is an r-row matrix that describes the r sensors by

$$\texttt{Sensor\_Spec} = \begin{bmatrix} \texttt{Sensor\_1} \\ \texttt{Sensor\_2} \\ \vdots \\ \texttt{Sensor\_r} \end{bmatrix}$$

Window 5.8. Function ssbeam2 (*Continued*)

where the jth row Sensor_j specifies the jth sensor according to Table 12.5.5; Actuator_Spec is an s-row matrix that describes the s actuators by

$$\texttt{Actuator\_Spec} = \begin{bmatrix} \texttt{Actuator\_1} \\ \texttt{Actuator\_2} \\ \vdots \\ \texttt{Actuator\_s} \end{bmatrix}$$

where the jth row Actuator_j specifies the jth actuator according to Table 12.5.6; and Dmp_Spec is a vector specifying modal damping according to the description given in Window 5.7 (for function trbeam2). If the beam has no damping, simply use [A, B, C] = ssbeam2(Sensor_Spec, Actuator_Spec).

Once matrices $[A]$, $[B]$, and $[C]$ of a state representation are obtained by function ssbeam2, the MATLAB Control System Toolbox can be used to study the properties of the control system (e.g., stability, controllability, and observability) and to design a feedback controller. Furthermore, functions respnubeam (Window 5.5) and nubmovie (Window 5.6) can be used to compute and animate the response of a nonuniform beam under feedback control.

EXAMPLE 5.4

The beam given in Example 5.1 is controlled via state feedback $\{u(t)\} = -[K]\{z(t)\}$ with a moment actuator located at the left end ($x = 0$), where $\{u(t)\}$ is the control input vector in Eq. (12.148) and $[K]$ is a gain matrix. The first three modes of the beam are considered in the control system formulation. Design the control gain matrix $[K]$ such that the closed-loop poles (eigenvalues) are $-1 \pm j4, -3 \pm j13, -5, -10$.

The control system design takes two steps. First, obtain a state-space model of the beam:

```
>> nonubeam([1 0.5 0.25], [4 10 0.25 1 0 -0.2], 2, [9 5 0 8 0.5 0
0.2 0], 3)
>> [A, B] = ssbeam2([1 2], [2 0])
A =
       0        0         0    1.0000         0         0
       0        0         0         0    1.0000         0
       0        0         0         0         0    1.0000
       0  10.9438  410.4158         0         0         0
       0 -17.6274 -111.2634         0         0         0
       0  -9.3902 -142.7942         0         0         0
B =
       0
       0
       0
  1.0000
 -3.1179
  5.7784
```

Second, design the feedback control gain by function `place` from the MATLAB Control System Toolbox:

```
>> p = [-1+4j -1-4j -3+13j -3-13j -5 -10];
>> K= place(A,B,p)
```

which gives the control gain matrix as follows

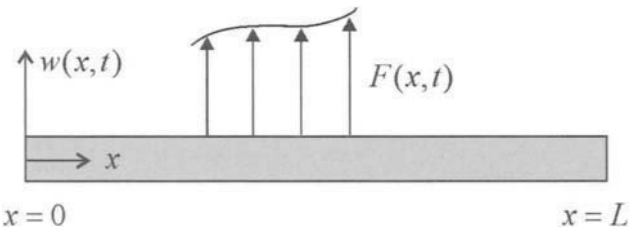
$$[K] = [3.4874 \quad -5.1373 \quad 34.1050 \quad 1.5741 \quad 0.4473 \quad 3.9493].$$

The closed-loop state matrix is

```
>> Ac = A-B*K
Ac =
         0         0         0    1.0000         0         0
         0         0         0         0    1.0000         0
         0         0         0         0         0    1.0000
   -3.4874   16.0811  376.3108   -1.5741   -0.4473   -3.9493
   10.8735  -33.6450   -4.9274    4.9078    1.3946   12.3134
  -20.1519   20.2955 -339.8675   -9.0957   -2.5846  -22.8205
```

12.6 Quick Solution Guide

Uniform Beams



Equation of Motion

$$\rho \frac{\partial^2}{\partial t^2} w(x,t) + d_v \frac{\partial}{\partial t} w(x,t) + EI \frac{\partial^4}{\partial x^4} w(x,t) = F(x,t)$$

Problems to Solve

(a) Eigenvalue problem (natural frequencies and mode shapes)

$$\rho \omega^2 u(x) = EI \frac{d^4}{dx^4} u(x), \quad 0 \le x \le L$$

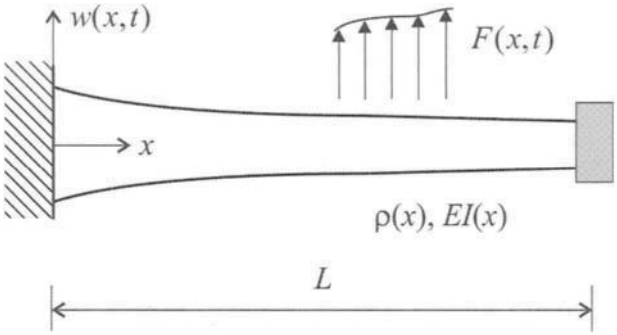
(b) Free vibration of a beam subject to initial disturbances ($F(x,t) = 0$)
(c) Forced vibration of a beam subject to an external force ($F(x,t) \neq 0$)

(d) Frequency response of a beam subject to a sinusoidal force

$$F(x,t) = f_0\, e^{j\omega t}\, \delta(x - x_f)$$

(d) Modeling and simulation for feedback control

Nonuniform Beams



Equation of Motion

$$\rho(x)\frac{\partial^2}{\partial t^2}w(x,t) + \frac{\partial^2}{\partial x^2}\left[EI(x)\frac{\partial^2}{\partial x^2}w(x,t)\right] = F(x,t)$$

Problems to Solve

(a) Eigenvalue problem (natural frequencies and mode shapes)

(b) Time response

(c) Control system formulation

MATLAB Functions

This chapter has a set (toolbox) of MATLAB functions for vibration analysis and feedback control of Euler-Bernoulli beams; see the table below. Refer to the corresponding window or section for the utility of each function. Usage of `tfbeam`, `clpbeam`, and `tfbeam2` requires the MATLAB Control System Toolbox.

	System Setup	
`setbeam`	Set up a uniform beam, and compute its natural frequencies and mode shapes	Window 2.1
`setdamp`	Specify or reset damping in a beam	Window 3.2
`systinfo`	Display the parameters, boundary conditions, damping status, natural frequencies, and mode shapes of a beam	Section 12.2.3
	Eigensolutions	
`eigenbeam`	Determine eigenvalues and normalized eigenfunctions of a beam	Window 2.2
`plotmsh`	Plot mode shape of a beam	Window 2.3
`nodalpts`	Determine the nodal points of an eigenfunction (mode shape)	Window 2.4
`animode`	Animate a mode of vibration of a beam	Window 2.5

(*continued*)

Dynamic Response		
`freebeam` `freebeam1`	Compute and display free response of a beam	Window 3.3
`forcedbeam` `forcedbeam1`	Compute and display forced response of a beam	Window 3.4
`plotbeam`	Plot time response of a beam at a spatial point	Window 3.6
`plotbeam1`	Plot spatial distribution of a beam response at a specific time	Window 3.6
`beammovie`	Animate time response of a beam	Window 3.7
`frfbeam`	Plot the magnitude and phase of steady-state response of a beam subject to a pointwise sinusoidal force	Window 3.8
Feedback Control		
`tfbeam`	Obtain the open-loop transfer function of a beam with a pair of sensor and actuator	Window 4.1
`ssbeam`	Obtain a state representation for a beam	Window 4.2
`clpbeam`	Obtain the closed-loop state equation for a beam under feedback control	Window 4.3
`contrbeam`	Compute the response of a beam with or without feedback control by a Runge-Kutta algorithm	Window 4.4
Nonuniform Beams		
`nonubeam`	Set up a nonuniform beam and compute its natural frequencies and mode shapes	Window 5.1
`systinfo2`	Display the parameters, boundary conditions, damping status, natural frequencies, and mode shapes of a nonuniform beam	Section 12.5.3
`plotmsh2`	Plot mode shape of a beam	Window 5.2
`animode2`	Animate a mode of vibration of a beam	Window 5.3
`getmkb0`	Create a discretized model for a beam	Window 5.4
`respnubeam`	Plot time response of a nonuniform beam	Window 5.5
`nubmovie`	Play animated vibration of a beam	Window 5.6
`tfbeam2`	Create the transfer function of a nonuniform beam under a pair of sensor and actuator	Window 5.7
`ssbeam2`	Create a state-space representation of a nonuniform beam under multiple sensors and actuators	Window 5.8
Utilities		
`TBdemo`	Show how the Toolbox works and what it can do	Section 12.1
`TBinfo`	Display the information about this Toolbox	Section 12.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 12.1

Solution Procedure

Dynamic analysis and feedback control of a uniform beam takes the following three steps:

Step 1. Set up system parameters, boundary conditions, and damping

```
>> setbeam(rou, EI, L, BC_Spec, Dmp_Spec)
```

Function `setbeam` also yields natural frequencies and mode shapes.

Step 2. Compute and display beam response

For free response at point `xr`,

```
>> y = freebeam(IC, xr);
```

For forced response at point `xr`,

```
>> y = forcedbeam(Load_Spec, xr);
```

For frequency response at point `xr` due to a sinusoidal load,

```
>> frfbeam(xr, xf, f0)
```

To animate beam response, type

```
>> beammovie(Load_Spec)
```

To plot beam response, type

```
>> plotbeam(y)
```

where `y` is obtained by `freebeam` or `forcedbeam`.

Step 3. Formulate control system

For a state representation,

```
>> ssbeam(Sensor_Spec, Actuator_Spec)
```

For a closed-loop state equation,

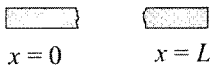
```
>> clpbeam(ID_control, K_gain, Gs)
```

For dynamic response of the controlled beam,

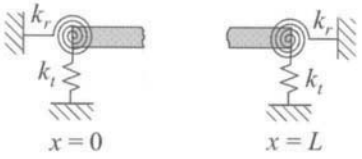
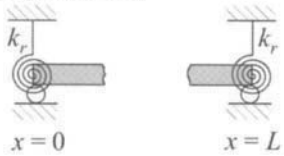
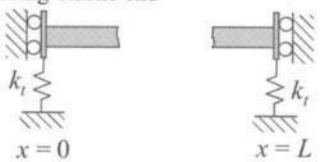
```
>> contrbeam(xr, Load_Spec, IC)
```

Note: `rou`, `EI`, and `L` are linear density, bending stiffness, and length of the beam; `BC_Spec` is a vector specifying beam boundary conditions; `Dmp_Spec` is a vector specifying damping in a beam; `IC` is a matrix specifying initial displacement and velocity; and `Load_Spec` is a vector specifying an external force. Also, `Sensor_Spec` and `Actuator_Spec` specify sensors and actuators, respectively; and `ID_control`, `K_gain`, and `Gs` specify a feedback control law. These arguments are explained as follows.

(1) Specification of boundary conditions by vector `BC_Spec` (as Table 12.2.3)

Boundary Condition *(k_r - torsional spring coefficient; k_t - translational spring coefficient)*	`BC_Spec = [BC_Left BC_Right]`	
	Left Boundary (x = 0) **`BC_Left`**	*Right Boundary (x = L)* **`BC_Right`**
(B1) Pinned or hinged end	[1]	[1]
(B2) Clamped or fixed end	[2]	[2]
(B3) Free end 	[3]	[3]

(continued)

Boundary Condition *(k_r - torsional spring coefficient;* *k_t - translational spring coefficient)*	**BC_Spec = [BC_Left BC_Right]** *Left Boundary (x = 0)* **BC_Left**	 *Right Boundary (x = L)* **BC_Right**
(B4) Sliding or guided end $x = 0$ $x = L$	[4]	[4]
(B5) Elastic-support  $x = 0$ $x = L$	[5 k_r k_t]	[5 k_r k_t]
(B6) Hinged-elastic end  $x = 0$ $x = L$	[6 k_r]	[6 k_r]
(B7) Sliding-elastic end  $x = 0$ $x = L$	[7 k_t]	[7 k_t]

(2) Specification of damping in a beam by vector `Dmp_Spec`

Vector:
`Dmp_Spec` (as an argument in functions `setbeam` and `setdamp`)

Purpose:

To specify damping in a beam.

Description:
Vector `Dmp_Spec` is used to specify the damping status of a beam with the following options:

(a) Viscous damping: `Dmp_Spec` = d_v, where d_v is a viscous damping coefficient.

(b) Modal damping: `Dmp_Spec` is a vector of the form

```
Dmp_Spec = [1 zeta_1, zeta_2 ... zeta_m]
```

in which $0 \leq$ `zeta_j` < 1 for all j. If $m < n$, where n is the total number of modes considered, the $(m+1)$th, ..., nth modes have the same damping ratio as `zeta_m`. For instance, `Damp_Spec` = [2 0.1 0.05] specifies a 10% damping ratio for the first mode, and a 5% damping ratio for the rest of the modes.

(c) No damping: `Dmp_Spec = [ ] or 0`

(3) Specification of initial disturbances by matrix `IC` (as in Window 3.3)

Matrix:
`IC` (as an argument in functions `freebeam` and `freebeam1`)

Purpose:

To specify initial disturbances of a beam, namely,

$$w(x,0) = a_0(x), \quad \frac{\partial}{\partial t} w(x,0) = b_0(x); \quad 0 \leq x \leq L$$

Description:
`IC` is a $2 \times np$ matrix specifying the initial displacement $a_0(x)$ and initial velocity $b_0(x)$ of a beam at np equally-spaced points along the beam:

$$\texttt{IC} = \begin{bmatrix} a_0(x_1) & a_0(x_2) & \cdots & a_0(x_{np}) \\ b_0(x_1) & b_0(x_2) & \cdots & b_0(x_{np}) \end{bmatrix}$$

where $x_k = k \times L/np$, $k = 1, 2, \ldots, np$, and L is the beam length.

(4) Load specification vector `Load_Spec` (as Table 12.3.2)

External Load	*F(x,t)*	`Load_Spec`
(L0) Pointwise impulse force	$I_0\delta(t)\delta(x - x_f)$	[0 x_f I_0]
(L1) Pointwise step force	$F_0\,\delta(x - x_f)$	[1 x_f F_0]
(L2) Pointwise sinusoidal force	$A\sin(\omega t + \phi_0)\delta(x - x_f)$	[2 x_f A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
(L10) Uniformly distributed impulse force	$I_0\delta(t)$, for $x_1 \leq x \leq x_1$	[10 x_1 x_2 I_0]
(L11) Uniformly distributed step force	F_0, for $x_1 \leq x \leq x_1$	[11 x_1 x_2 F_0]
(L12) Uniformly distributed sinusoidal force	$A\sin(\omega t + \phi_0)$, for $x_1 \leq x \leq x_1$	[12 x_1 x_2 A ω ϕ_0] ω in rad/s, ϕ_0 in degrees

(*continued*)

External Load	*F(x,t)*	Load_Spec
(L20) Pointwise impulse torque	$-I_0\delta(t)\dfrac{d}{dx}\delta(x-x_\tau)$	[20 x_τ I_0]
(L21) Pointwise step torque	$-\tau_0\dfrac{d}{dx}\delta(x-x_\tau)$	[21 x_τ τ_0]
(L22) Pointwise sinusoidal torque	$-A\sin(\omega t+\phi_0)\dfrac{d}{dx}\delta(x-x_\tau)$	[22 x_τ A ω ϕ_0] ω in rad/s, ϕ_0 in degrees

(5) Specification of sensors and actuators by Sensor_Spec and Actuator_Spec

Assume that the beam is controlled by r sensors located at $s_j, j = 1, 2, \ldots, r$, and s actuators located at $a_k, k = 1, 2, \ldots, s$. The sensors are described by an r-row matrix

$$\texttt{Sensor\_Spec} = \begin{bmatrix} \texttt{Sensor\_1} \\ \texttt{Sensor\_2} \\ \vdots \\ \texttt{Sensor\_r} \end{bmatrix}$$

where the jth row Sensor_j specifies the jth sensor according to the following table.

Sensor Type	Sensor_j
(S1) Transverse displacement $y_j(t) = w(s_j, t)$	[1 s_j]
(S2) Transverse velocity $y_j(t) = \dfrac{\partial}{\partial t}w(s_j, t)$	[2 s_j]
(S3) Slope or rotation $y_j(t) = \theta(s_j, t) = \dfrac{\partial}{\partial x}w(s_j, t)$	[3 s_j]
(S4) Rotation velocity $y_j(t) = \dfrac{\partial}{\partial t}\theta(s_j, t)$	[4 s_j]
(S5) Strain $y_j(t) = \dfrac{\partial^2}{\partial x^2}w(s_j, t) = \kappa(s_j, t)$	[5 s_j]
(S6) Rate of change of strain $y_j(t) = \dfrac{\partial}{\partial t}\kappa(s_j, t)$	[6 s_j]

The actuators are described by an s-row matrix

$$\texttt{Actuator\_Spec} = \begin{bmatrix} \texttt{Actuator\_1} \\ \texttt{Actuator\_2} \\ \vdots \\ \texttt{Actuator\_s} \end{bmatrix}$$

where the jth row `Actuator_j` specifies the jth actuator according to the following table.

Actuator Type	`Actuator_j`
(A1) Transverse force $F_{c,j}(x,t) = f_j(t)\delta(x - a_j)$	`[1 aj]`
(A2) Moment $F_{c,j}(x,t) = -f_j(t)\dfrac{d}{dx}\delta(x - a_j)$	`[2 aj]`

(6) Specification of feedback control law by `ID_control`, `K_gain`, and `Gs`

`ID_control` is an integer identifying one of the three feedback controllers considered in this Toolbox, and `K_gain` and `Gs` are gain matrices, as explained below:

`ID_control` = 1: state feedback $\{U(t)\} = -[K_c]\{z(t)\}$, where `K_gain` is the gain matrix $[K_c]$ and `Gs` is not assigned.

`ID_control` = 2: output feedback $\{U(t)\} = -[K_c]\{y(t)\}$, where `K_gain` is the gain matrix $[K_c]$ and `Gs` is the matrix $[g_s]$ in Eq. (12.81).

`ID_control` = 3: PID feedback, where `K_gain = [Kp Ki Kd]` with `Kp`, `Ki`, and `Kd` being the controller parameters and `Gs` is not assigned.

`ID_control` = 0: no feedback control.

12.7 References

References on Vibrations of Euler-Bernoulli Beams

1. Balachandran B, Magrab E B 2003 *Vibrations*, Brooks/Cole Publishing Co.: New York.
2. Den Hartog J P 1985 *Mechanical Vibrations*, Dover Publications: New York.
3. Inman D J 2000 *Engineering Vibrations*, 2nd edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
4. Meirovitch L 1967 *Analytical Methods in Vibrations*, Macmillan: New York.
5. Rao S S 2003 *Mechanical Vibrations*, 4th edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
6. Timoshenko S 1990 *Vibration Problems in Engineering*, 5th ed.: John Wiley & Sons.

References on Feedback Control

7. Chen C-T 1998 *Linear System Theory and Design*, 3rd edition, Oxford University Press: New York.
8. Dorf R C, Bishop R H 2000 *Modern Control Systems*, 9th edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
9. Kuo B C, Golnaraghi F 2002 Automatic Control Systems, 8th edition, Wiley Text Books.
10. Nise N S 2000 Control Systems Engineering, 3rd edition, Wiley Text Books.
11. Ogata K 2001 *Modern Control Engineering*, 4th edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.

This Page Intentionally Left Blank

13 Dynamic Analysis of Bars, Shafts, and Strings

Inside
- Getting Started
- System Description
- Dynamic Response
- Free Vibration of Stepped Systems
- Quick Solution Guide
- References

13.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for dynamic analysis of elastic bars, circular shafts, and taut strings.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on the computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

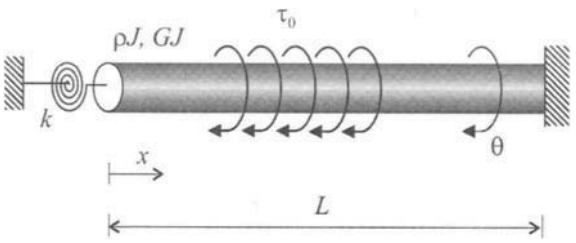


FIGURE 13.1.1 A circular shaft with an end spring.

(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox is properly installed, and it is ready for use.

Quick Tutorial

To show how to use the Toolbox, consider a circular shaft in Fig. 13.1.1, which is connected to a torsional spring of coefficient k at the left end and fixed at the right end. The shaft is subject to a uniformly distributed step torque $\tau(x,t) = \tau_0$ in region $a \le x \le b$. Let the parameters of the shaft and torque be

$$\rho_v J = 18.6 \text{ kg} \cdot \text{m}, \quad GJ = 245 \text{ N} \cdot \text{m}^2, \quad L = 5 \text{ m}, \quad k = 210 \text{ N} \cdot \text{m/rad}$$
$$\tau_0 = 25 \text{ N}, \quad a = 2.5 \text{ m}, \quad b = 3.5 \text{ m}$$

To determine the time history of the shaft rotation θ at $x = 0.5$ m, in the MATLAB command window, type the following:

```
>> rou = 18.6; GJ = 245; L = 5; k = 210;
>> setshaft(rou, GJ, L, [3 k 1])
>> load = [10 2.5 3.5 25]; xr = 0.5
>> forced2os(load, xr)
```

where >> is the prompt in the MATLAB command window. This yields the natural frequencies of the shaft as below

```
    wn        fn
  1.8849    0.3000
  3.9085    0.6221
  6.0459    0.9622
  8.2432    1.3120
 10.4713    1.6666
 12.7162    2.0238
 14.9712    2.3827
 17.2325    2.7426

where
wn = natural frequencies in rad/sec
fn = wn/(2*pi), natural frequencies in Hertz
```

and the time response plotted in Fig. 13.1.2.

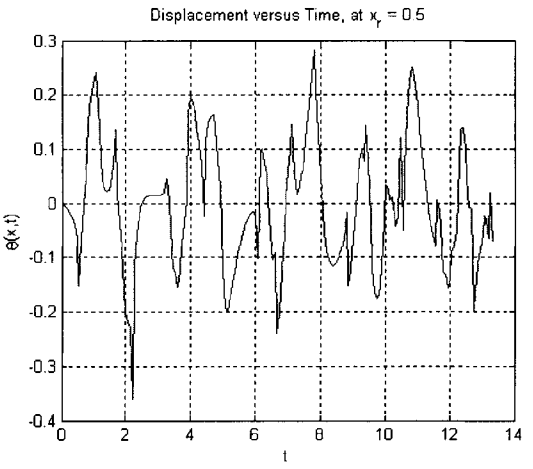


FIGURE 13.1.2

In the above dynamic analysis, two functions from the Toolbox have been called:

(a) Function **setshaft** that inputs the shaft parameters, sets up the boundary conditions, and computes the natural frequencies of the shaft;
(b) Function **forced2os** that computes the response of the shaft under the external torque.

These functions, along with others from the Toolbox, will be presented in sequel.

For a quick solution, the user may refer directly to the Quick Solution Guide (Section 5) or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To better understand how the toolbox works, the user is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 13.6.

13.2 System Description

In this section, the governing equations and eigenvalue problem of three types of distributed dynamic systems are presented; the related MATLAB functions are introduced.

13.2.1 Governing Equations

Described in Tables 13.2.1 to 13.2.3 are three types of uniform distributed systems:

- Elastic bars in longitudinal vibration;
- Circular shafts in torsional vibration; and
- Taut strings in lateral vibration.

Equation of Motion The equations of motion in Tables 13.2.1 to 13.2.3 are in a unified form (References 1 and 3):

$$\rho \frac{\partial^2}{\partial t^2} w(x,t) - \alpha_{ST} \frac{\partial^2}{\partial x^2} w(x,t) = F(x,t), \quad 0 \le x \le L \tag{13.1}$$

TABLE 13.2.1 Elastic Bars in Longitudinal Vibration

Governing equation

$$\rho_v A \frac{\partial^2}{\partial t^2} u(x,t) - EA \frac{\partial^2}{\partial x^2} u(x,t) = q(x,t)$$

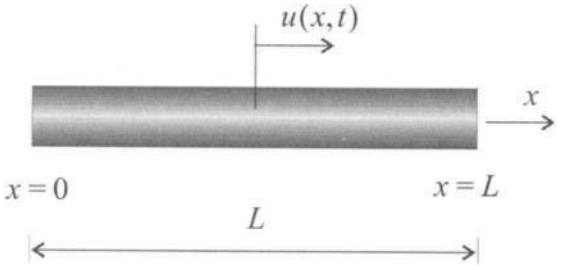
$u(x,t)$: longitudinal displacement

$q(x,t)$: external load

ρ_v : material density (mass per unit volume)

E : Young's modulus

A : cross-section area



Internal force (axial force)

$$N(x,t) = EA \frac{\partial}{\partial x} u(x,t)$$

EA: longitudinal rigidity

Sign convention

Boundary conditions

(B1) Fixed end

Left: $u(0,t) = 0$

Right: $u(L,t) = 0$

(B2) Free end

Left: $\frac{\partial}{\partial x} u(0,t) = 0$

Right: $\frac{\partial}{\partial x} u(L,t) = 0$

(B3) End spring

Left: $ku(0,t) - EA \frac{\partial}{\partial x} u(0,t) = 0$

Right: $ku(L,t) + EA \frac{\partial}{\partial x} u(L,t) = 0$

TABLE 13.2.2 Circular Shafts in Torsional Vibration

Governing equation

$$\rho_v J \frac{\partial^2}{\partial t^2} \theta(x,t) - GJ \frac{\partial^2}{\partial x^2} \theta(x,t) = \tau(x,t)$$

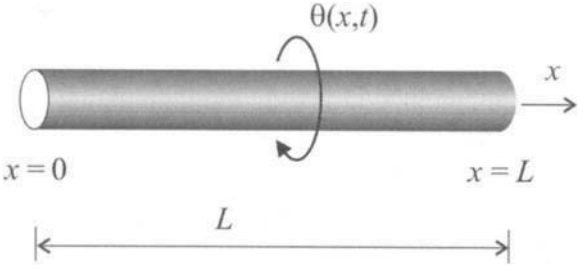
$\theta(x,t)$: rotation (twist)

$\tau(x,t)$: external torque

ρ_v : material density (mass per unit volume)

G : shear modulus

J : polar moment of inertia



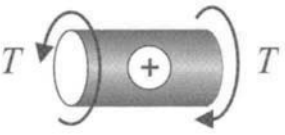
(continued)

TABLE 13.2.2 Circular Shafts in Torsional Vibration (*Continued*)

Internal force (torque)

$T(x,t) = GJ\dfrac{\partial}{\partial x}\theta(x,t)$

GJ: torsional rigidity



Sign convention

Boundary conditions and boundary disturbances

(B1) Fixed end	(B2) Free end	(B3) End spring
x = 0 x = L	$x = 0$ $x = L$	$x = 0$ $x = L$
Left: $\theta(0,t) = 0$	Left: $\dfrac{\partial}{\partial x}\theta(0,t) = 0$	Left: $k\theta(0,t) - GJ\dfrac{\partial}{\partial x}\theta(0,t) = 0$
Right: $\theta(L,t) = 0$	Right: $\dfrac{\partial}{\partial x}\theta(L,t) = 0$	Right: $k\theta(L,t) + GJ\dfrac{\partial}{\partial x}\theta(L,t) = 0$

TABLE 13.2.3 Taut Strings in Transverse Vibration

Governing equation

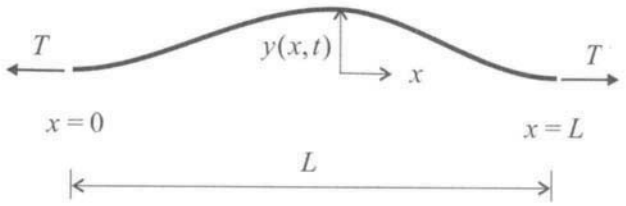
$\mu\dfrac{\partial^2}{\partial t^2}y(x,t) - T\dfrac{\partial^2}{\partial x^2}y(x,t) = p(x,t)$

$y(x,t)$: lateral displacement

$p(x,t)$: lateral external load

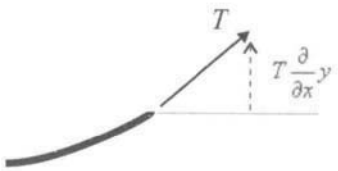
μ : linear density (mass per unit length)

T : string tension



Internal force (lateral projection of tension)

$T_y = T\dfrac{\partial}{\partial x}y(x,t)$



Boundary conditions and boundary disturbances

(B1) Fixed end	(B2) Free end	(B3) End spring
$x = 0$ $x = L$	$x = 0$ $x = L$	$x = 0$ $x = L$
Left: $y(0,t) = 0$	Left: $\dfrac{\partial}{\partial x}y(0,t) = 0$	Left: $ky(0,t) - T\dfrac{\partial}{\partial x}y(0,t) = 0$
Right: $y(L,t) = 0$	Right: $\dfrac{\partial}{\partial x}y(L,t) = 0$	Right: $ky(L,t) + T\dfrac{\partial}{\partial x}y(L,t) = 0$

where $w(x,t)$ is a generalized displacement; $F(x,t)$ is an external load; and ρ and α_{ST} are the corresponding inertia and stiffness (rigidity) parameters. For instance, for a bar, $w(x,t) = u(x,t)$, $\rho = \rho_v A$, and $\alpha_{ST} = EA$. These continua shall be called second-order distributed dynamic systems because Eq. (13.1) involves second-order spatial derivative.

Internal Force

$$\sigma(x,t) = \alpha_{ST}\frac{\partial}{\partial x}w(x,t) \tag{13.2}$$

Here $\sigma(x)$ is N for a bar, T for a shaft, and T_y for a string.

Boundary Conditions
(B1) Fixed end:

$$\begin{aligned} &\text{left end (at } x = 0), && w(0,t) = 0 \\ &\text{right end (at } x = L), && w(L,t) = 0 \end{aligned} \tag{13.3a}$$

(B2) Free end:

$$\begin{aligned} &\text{left end (at } x = 0), && \frac{\partial}{\partial x}w(0,t) = 0 \\ &\text{right end (at } x = L), && \frac{\partial}{\partial x}w(L,t) = 0 \end{aligned} \tag{13.3b}$$

(B3) End spring:

$$\begin{aligned} &\text{left end (at } x = 0), && k_s w(0,t) - \alpha_{ST}\frac{\partial}{\partial x}w(0,t) = 0 \\ &\text{right end (at } x = L), && k_s w(L,t) + \alpha_{ST}\frac{\partial}{\partial x}w(L,t) = 0 \end{aligned} \tag{13.3c}$$

where k_s is a spring coefficient.

13.2.2 Eigenvalue Problem

The eigenvalue problem of a second-order distributed dynamic system is derived by assuming its displacement as

$$w(x,t) = v(x)e^{j\omega t}, \quad j = \sqrt{-1} \tag{13.4}$$

and substituting it into Eq. (13.1) (with $F(x,t) = 0$). This yields the eigenequation

$$\omega^2 v(x) + c^2\frac{d^2}{dx^2}v(x) = 0, \quad 0 \le x \le L \tag{13.5}$$

where $c = \sqrt{\alpha_{ST}/\rho}$, ω is the eigenvalue, and $v(x)$ is the eigenfunction. Physically, ω is a natural frequency of the system, while $v(x)$ is the associate mode shape. Equation (13.5) has

an infinite number of solutions: ω_k, $u_k(x)$, $k = 1, 2, 3, \ldots$ The eigenvalues are nonnegative, and can be arranged in an ascending order

$$0 \leq \omega_1 < \omega_2 < \omega_3 < \cdots \tag{13.6}$$

The eigenfunctions are in the orthonormal relations

$$\begin{aligned} \rho \int_0^L v_j(x) v_k(x)\, dx &= \delta_{jk} \\ -\alpha_{ST} \int_0^L v_j(x) \frac{d^2}{dx^2} v_k(x)\, dx &= \delta_{jk} \omega_k^2 \end{aligned} \tag{13.7}$$

where the delta function δ_{jk} is one for $j = k$, and zero for $j \neq k$.

For a nonzero eigenvalue ω (flexible mode), the associate eigenfunction is given by

$$v(x) = a_1 \cos \beta x + a_2 \sin \beta x \tag{13.8}$$

where

$$\beta = \frac{\omega}{c} = \omega \sqrt{\frac{\rho}{\alpha_{ST}}} \tag{13.9}$$

and parameters a_1 and a_2 are determined by the boundary conditions (13.3). For a zero eigenvalue (rigid-body mode), which only exists with free-free (B2-B2) boundary conditions, the associate eigenfunction is constant; namely,

$$v(x) = a \tag{13.10}$$

To solve the eigenvalue problem, substitute Eq. (13.8) into the boundary conditions of the system, yielding a homogenous equation

$$[A(\mu)]\,\{a\} = 0 \tag{13.11}$$

where $\{a\} = \begin{pmatrix} a_1 & a_2 \end{pmatrix}^T$, μ is a nondimensional eigenvalue given by

$$\mu = \beta L = \frac{\omega}{c} L \tag{13.12}$$

and $[A(\mu)]$ is a 2-by-2 matrix with elements as functions of μ. For a nonzero solution of $\{a\}$, the determinant of $[A(\mu)]$ must be zero, rendering the characteristic equation

$$\Delta(\mu) = \det [A(\mu)] = 0 \tag{13.13}$$

Function $\Delta(\mu)$ has an infinite number of roots μ_k, $k = 1, 2, \ldots$, which are related to the natural frequencies ω_k by

$$\omega_k = \frac{\mu_k c}{L} = \beta_k c \tag{13.14}$$

With a known μ_k, a nonzero $\{a\}$ from $[A(\mu_k)]\,\{a\} = 0$ gives the associate modal shape $v_k(x)$ by Eq. (13.8).

As an example, consider a second-order distributed system with fixed-free ends, whose eigenfunctions satisfy the boundary conditions $v(0) = 0$ and $\frac{d}{dx}v(L) = 0$. Substituting Eq. (13.8) into the boundary conditions gives

$$\begin{bmatrix} 1 & 0 \\ -\mu \sin \mu & \mu \cos \mu \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = 0$$

which, by Eq. (13.13), leads to the characteristic equation

$$\cos \mu = 0.$$

Thus, the nondimensional eigenvalues are

$$\mu_k = \beta_k L = (k - 1/2)\pi, \quad k = 1, 2, 3, \ldots$$

and the associate eigenfunctions are

$$v_k(x) = \sqrt{\frac{2}{\rho L}} \sin \beta_k x, \quad k = 1, 2, 3, \ldots$$

which satisfy the normalization condition $\rho \int_0^L v_k^2(x)\, dx = 1$.

Table 13.2.4 lists the characteristic equations and eigenfunctions of second-order distributed systems for the three types of boundary conditions described in Eqs. (13.3).

13.2.3 System Setup by the Toolbox

For dynamic analysis of second-order distributed systems, the Toolbox of this chapter provides functions `setbar`, `setshaft`, and `setstring`; see Window 3.1.

Window 2.1. Functions `setbar`, `setshaft`, `setstring`

MATLAB Functions: **`setbar`, `setshaft`, `setstring`**

Purpose:
To set up a second-order uniformly distributed dynamic system and to compute its eigensolutions.

Synopsis:

```
setbar(rou, EA, L, BC_Spec, n)
setshaft(rou, GJ, L, BC_Spec, n)
setstring(rou, T, L, BC_Spec, n)
```

Description:
`setbar(rou, EA, L, BC_Spec)` undertakes the following tasks:

(a) It inputs linear density `rou`, longitudinal rigidity `EA`, and bar length `L`;
(b) It sets up boundary conditions by vector `BC_Spec` (see Table 13.2.5); and
(c) It computes natural frequencies and normalized mode shapes.

`setbar(rou, EA, L, BC_Spec, n)` selects the number `n` of modes (eigenfunctions) for dynamic analysis. By default, `n = 200`.

Functions `setshaft` and `setstring` undertake similar tasks, for a shaft and a string, respectively.

TABLE 13.2.4 Characteristic Equation and Normalized Eigenfunction

Characteristic Equation	*Eigenfunctions $v_k(x)$ **
Fixed-fixed (B1-B1): $\sin\mu = 0$ with roots $\mu_k = k\pi$ for $k \geq 1$	$\sqrt{\frac{2}{\rho L}}\sin\beta_k x$
Fixed-free (B1-B2): $\cos\mu = 0$ with roots $\mu_k = (k-\frac{1}{2})\pi$ for $k \geq 1$	$\sqrt{\frac{2}{\rho L}}\sin\beta_k x$
Fixed-end spring (B1-B3): $\cos\mu + \frac{k_s L}{\alpha_{ST}}\frac{1}{\mu}\sin\mu = 0$ with k_s - spring coefficient	$\sqrt{\frac{2}{\rho L\alpha_k}}\sin\beta_k x$ with $\alpha_k = 1 - \frac{1}{2\mu_k}\sin 2\mu_k$
Free-free (B2-B2): $\mu\sin\mu = 0$ with roots $\mu = 0$ and $\mu_k = k\pi$ for $k \geq 1$	Rigid-body mode (for $\mu = 0$): $v(x) = 1/\sqrt{\rho L}$ Flexible modes (for $\mu_k = k\pi$): $\sqrt{\frac{2}{\rho L}}\cos\beta_k x$
Free-end spring (B2-B3) $\sin\mu - \frac{k_s L}{\alpha_{ST}}\frac{1}{\mu}\cos\mu = 0$ with k_s - spring coefficient	$\sqrt{\frac{2}{\rho L\alpha_k}}\cos\beta_k x$ with $\alpha_k = 1 + \frac{1}{2\mu_k}\sin 2\mu_k$
End spring-end spring (B3-B3) $\left(1 - \frac{k_1 k_2 L^2}{\alpha_{ST}^2\mu^2}\right)\sin\mu = \frac{(k_1+k_2)L}{\alpha_{ST}\mu}\cos\mu$ with k_1 - coefficient of the left spring k_2 - coefficient of the right spring	$\sqrt{\frac{2}{\rho L\alpha_k(1+\gamma_k^2)}}(\cos\beta_k x + \gamma_k\sin\beta_k x)$ with $\alpha_k = 1 + \frac{\sin(\mu_k - \theta_k) + \sin\theta_k}{2\mu_k}$ $\gamma_k = \frac{k_1 L}{\alpha_{ST}\mu_k}$, $\theta_k = \tan^{-1}\gamma_k$

* The eigenfunctions are normalized by $\rho\int_0^L v_k^2(x)dx = 1$ and $\mu_k = \beta_k L$ is a nondimensional eigenvalue.

TABLE 13.2.5 Specification of Boundary Conditions

Boundary Conditions	**BC_Spec = [BC_Left BC_Right]**	
	Left Boundary (x = 0) `BC_Left`	*Right Boundary (x = L)* `BC_Right`
(B1) Fixed end At $x = 0$: $w(0,t) = 0$ At $x = L$: $w(L,t) = 0$	[1]	[1]
(B2) Free end At $x = 0$: $\frac{\partial}{\partial x}w(0,t) = 0$ At $x = L$: $\frac{\partial}{\partial x}w(L,t) = 0$	[2]	[2]
(B3) End spring At $x = 0$: $k_s w(0,t) - \alpha_{ST}\frac{\partial}{\partial x}w(0,t) = 0$ At $x = L$: $k_s w(L,t) + \alpha_{ST}\frac{\partial}{\partial x}w(L,t) = 0$ (k_s - spring coefficient)	[3 k_s]	[3 k_s]

Note 1. Functions `setbar`, `setshaft`, and `setstring` must be executed before other functions of dynamic analysis can be used. However, these functions need to be called once for a system in different cases of initial and external disturbances.

Note 2. Functions `setbar`, `setshaft`, and `setstring` display computed system eigenvalues by the following four parameters:

ω_k, natural frequencies in rad/s

$f_k = \omega_k/2\pi$, natural frequencies in Hertz

$\beta_k = \omega_k \sqrt{\rho/\alpha_{ST}}$

$\mu_k = \beta_k L$, nondimensional eigenvalues

Note 3. The boundary specification vector `BC_Spec` mentioned in Table 13.2.5 has the form

```
BC_Spec = [BC_Left BC_Right]
```

where the subvectors `BC_Left` and `BC_Right` describe the boundary conditions at the left and right ends, respectively. The form of `BC_Spec` is illustrated on the following three configurations of second-order distributed systems:

(a) Fixed-fixed (B1-B1) boundaries:

```
BC_Spec = [1 1]
```

(b) Free-fixed (B2-B1) boundaries:

```
BC_Spec = [2 1]
```

(c) Fixed-end spring (B1-B3) boundaries:

```
BC_Spec = [1 3 ks]
```

where `ks` is the coefficient of the spring at the right end ($x = L$).

Note 4. To get access to the information of system parameters, boundary conditions, and the eigensolutions given by `setbar` or `setshaft` or `setstring`, call function **`systinfo`** at any time:

```
>>systinfo
```

EXAMPLE 2.1

A circular shaft in torsional vibration is fixed at the left end, and supported by a torsional spring of coefficient $k = 20$ at the right end. The shaft parameters are $\rho_v J = 2$, $GJ = 100$, and $L = 1.6$. To set up the system and to obtain system eigensolutions, execute the following commands

```
>> rou = 2; GJ = 100; L = 1.6;
>> BC_Spec = [1 3 20];
>> setshaft(rou, GJ, L, BC_Spec)
```

which yield the following eigensolutions of the shaft

```
   wn          fn                    Flexible Modes: v(x)=a1*cos(beta*x)+a2*sin(beta*x)
  7.7406     1.2320                   Mode No.    beta        a1        a2
 21.1215     3.3616                   1.0000     1.0947       0      0.7535
 34.8891     5.5528                   2.0000     2.9870       0      0.7851
 48.7223     7.7544                   3.0000     4.9341       0      0.7886
 62.5779     9.9596                   4.0000     6.8904       0      0.7895
 76.4438    12.1664                   5.0000     8.8499       0      0.7899
 90.3153    14.3741                   6.0000    10.8108       0      0.7901
104.1901    16.5824                   7.0000    12.7725       0      0.7903
wn = natural frequencies in rad/s     8.0000    14.7347       0      0.7903
fn = natural frequencies in Hertz
```

and plot the spatial distributions of the first four mode shapes in Fig. 13.2.1.

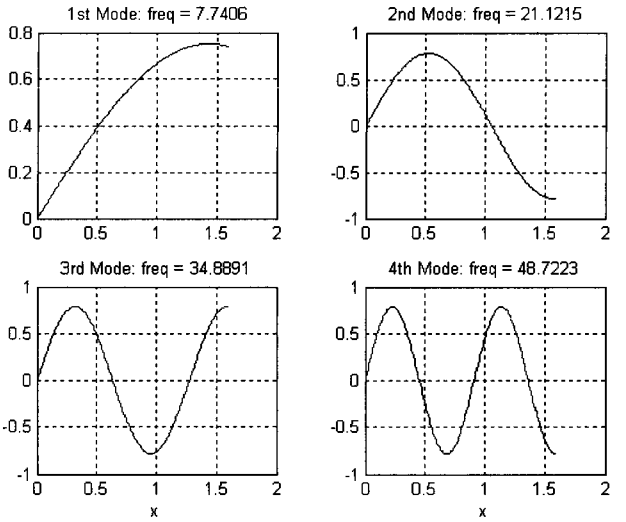


FIGURE 13.2.1

13.2.4 Display of Modes of Vibration

Besides the functions in Window 2.1, the Toolbox has four other functions, `eigen2os`, `nodalpts`, `plotmsh`, and `animode`, which are useful for displaying and animating modes of vibrations of a second-order distributed system; see Windows 2.2 to 2.5.

Window 2.2. Function `eigen2os`

MATLAB Function: **`eigen2os`**

Purpose:
To determine the eigenvalues and normalized eigenfunctions of a second-order distributed dynamic system.

Window 2.2. Function `eigen2os` (*Continued*)

Synopsis:

```
[y, msh] = eigen2os(n)
```

Description:

`[y, msh] = eigen2os(n)` returns the first `n` eigenvalues of the system in a vector `y` and the coefficients of normalized eigenfunctions in a 3-by-n matrix `msh`. The kth column of the matrix `msh` contains the parameters of the kth eigenfunction as follows

for a flexible mode $v(x) = a_1 \cos \beta x + a_2 \sin \beta x$

`msh(1,k)` = β, `msh(2,k)` = a_1, `msh(3,k)` = a_2;

for a rigid-body mode $v(x) = a$

`msh(1,k) = 0, msh(2,k) = a, msh(3,k) = 0.`

Window 2.3. Function `nodalpts`

MATLAB Function: **`nodalpts`**

Purpose:

To determine the nodal points of an eigenfunction. A nodal point of an eigenfunction $v(x)$ is a point x_* between 0 and L such that $v(x_*) = 0$.

Synopsis:

```
nodalpts(n)
y = nodalpts(n)
```

Description:

`nodalpts(n)` determines the nodal points of the nth mode shape of a system
`y = nodalpts(n)` returns the computed nodal points in a vector `y`.

Window 2.4. Function `plotmsh`

MATLAB Function: **`plotmsh`**

Purpose:

To plot mode shape of a second-order distributed dynamic system.

Synopsis:

```
plotmsh(k)
plotmsh(k, npts)
[y, x] = plotmsh(k, npts)
```

Window 2.4. Function `plotmsh` (*Continued*)

Description:
`plotmsh(k)` plots the *k*th mode shape of a system over its spatial region [0, L].

`plotmsh(k, npts)` plots the *k*th mode shape at `npts` equally-spaced points in the region [0, L]. By default, `npts` = 201.

`[y, x] = plotmsh(k npts)` returns the computed mode shape in a vector `y` and spatial points of computation in a vector `x`. The data in `x` and `y` can be used to plot the mode shape by the MATLAB function `plot(x, y)`.

Window 2.5. Function `animode`

MATLAB Function: **`animode`**

Purpose:
To animate modes of vibration of a second-order distributed dynamic system in its spatial region.

Synopsis:

```
animode(k)
animode(k, tf, n_frames, IS_control)
F = animode(k, tf, n_frames, IS_control)
```

Description:
`animode(k)` animates the *k*th mode of vibration of the system.

`animode(k, tf, n_frames, IS_control)` plays the animation of the *k*th mode of vibration with specified parameters, where `tf` is total animation time, `n_frames` is the number of frames in the animation, and `IS_control` is an animation control parameter with the options

`IS_control = 0`	play frames continuously
`IS_control = 1`	play frames one by one

By default, `tf` is eight times the period ($1/\omega_k$) of the mode, `n_frames` = 97, and `IS_control = 0`. Each of `tf`, `n_frames`, and `IS_control` can be replaced by `[ ]` as a placeholder for its default value.

`F = animode(k, tf, n_frames)` returns a set of frames on the time response of the *k*th mode in a matrix `F`, which can be played back by the MATLAB function `movie(F)`.

EXAMPLE 2.2

For the circular shaft considered in Example 2.1, the 1st, 3rd, and 7th mode shapes are plotted in one figure by

```
>> setshaft(2, 100, 1.6, [1 3 20])
>> [y1, x] = plotmsh(1);
```

```
>> y2 = plotmsh(3);
>> y3 = plotmsh(7);
>> plot(x, y1, x, y2, x, y3);
>> grid; xlabel('x')
```

which yields Fig. 13.2.2. Also,

```
>> eigen2os(12)
```

gives the first 12 natural frequencies: 7.7406, 21.1215, 34.8891, 48.7223, 62.5779, 76.4438, 90.3153, 104.1901, 118.0670, 131.9455, 145.8250, 159.7052, which are in rad/s.

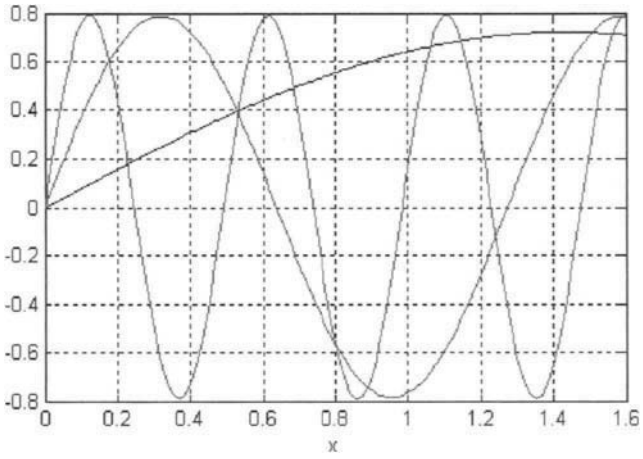


FIGURE 13.2.2

EXAMPLE 2.3

Consider again the shaft given in Example 2.1. By

```
>> setshaft(2, 100, 1.6, [1 3 20])
>> animode(3, 2*pi/34.8891, 7, 1)
```

the third mode of vibration of the shaft is animated. Shown in Fig. 13.2.3 are six frames of animation at times $t_i = i \times (2\pi/\omega_3)/6$, $i = 0, 1, \ldots, 5$, where $\omega_3 = 34.8891$ rad/s is the third natural frequency of the shaft.

13.3 Dynamic Response

In this section, the dynamic response of a second-order distributed system is obtained by modal expansion; the related MATLAB functions from the Toolbox are presented.

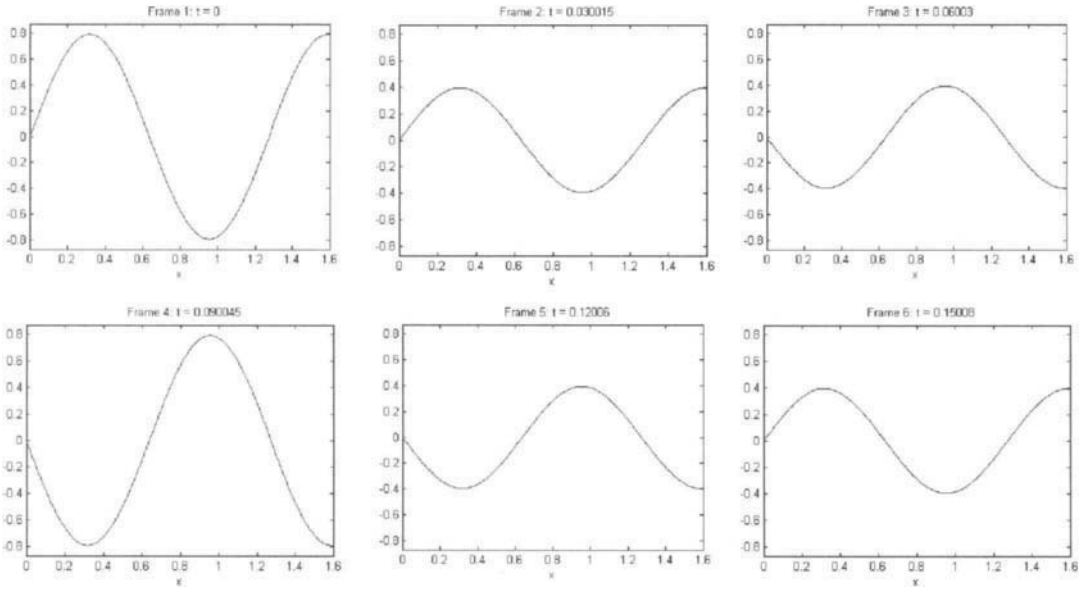


FIGURE 13.2.3 Frames of the animated vibration of the shaft in Example 2.3.

13.3.1 Modal Expansion

A fundamental problem in dynamic analysis of a second-order distributed dynamic system is to solve the equation of motion

$$\rho \frac{\partial^2}{\partial t^2} w(x,t) - \alpha_{ST} \frac{\partial^2}{\partial x^2} w(x,t) = F(x,t) = D(x)f(t), \quad 0 \le x \le L \tag{13.15}$$

subject to the boundary conditions specified by Eqs. (13.3) and the initial conditions

$$w(x,0) = a_0(x), \quad \frac{\partial}{\partial t} w(x,0) = b_0(x), \quad 0 \le x \le L \tag{13.16}$$

Here $D(x)$ describes the spatial distribution of the external force, and $a_0(x)$ and $b_0(x)$ are the initial displacement and velocity, respectively. To determine $w(x,t)$ in Eq. (13.15), express it by the series

$$w(x,t) = \sum_{k=1}^{\infty} v_k(x) q_k(t) \tag{13.17}$$

where $v_k(x)$ are the system eigenfunctions obtained from Eq. (13.5) of Section 13.2.2 and $q_k(t)$ are unknown modal coordinates to be determined. Substituting Eq. (13.17) into Eq. (13.15) gives

$$\sum_{k=1}^{\infty} \rho v_k(x) \left\{ \ddot{q}_k(t) + \omega_k^2 q_k(t) \right\} = D(x)f(t) \tag{13.18}$$

Multiplying Eq. (13.18) by $v_j(x)$, integrating the resulting equation over the spatial domain $0 \le x \le L$ and making use of the orthonormal conditions (13.7), yields the differential

equations governing the modal coordinates

$$\ddot{q}_k(t) + \omega_k^2 q_k(t) = b_k f(t), \quad k = 1, 2, \ldots \tag{13.19}$$

where

$$b_k = \int_0^L D(x) v_k(x)\, dx \tag{13.20}$$

Also, the initial conditions given in Eq. (13.16) are converted to

$$q_k(0) = \rho \int_0^L a_0(x)\, v_k(x)\, dx, \quad \dot{q}_k(0) = \rho \int_0^L b_0(x)\, v_k(x)\, dx \tag{13.21}$$

Solve Eq. (13.19) for $q_k(t)$ to obtain the dynamic response $w(x,t)$ in the series (13.17). This process is called *modal analysis* or *eigenfunction expansion*. In numerical simulation, an N-term truncation of the series is needed:

$$w(x,t) \approx w_N(x,t) = \sum_{k=1}^{N} v_k(x) q_k(t) \tag{13.22}$$

13.3.2 Free Vibration

For a second-order system in free vibration, the modal coordinates are governed by

$$\ddot{q}_k(t) + \omega_k^2 q_k(t) = 0, \quad k = 1, 2, \ldots \tag{13.23}$$

with the initial conditions given in Eq. (13.21). The solution of Eq. (13.23) is given in the following two cases:

for $\omega_k \neq 0$ (flexible mode)

$$q_k(t) = q_k(0) \cos \omega_k t + \dot{q}_k(0) \frac{1}{\omega_k} \sin \omega_k t \tag{13.24}$$

for $\omega_k = 0$ (rigid-body mode)

$$q_k(t) = q_k(0) + \dot{q}_k(0)\, t \tag{13.25}$$

The Toolbox of this chapter provides functions `free2os` and `free2os1` for computing free vibration of second-order distributed systems; see Window 3.1. In `free2os` and `free2os1`, the initial values of the modal coordinates are estimated by numerical integration of Eq. (13.21)

$$q_k(0) \approx \frac{\rho L}{np} \sum_{j=1}^{np} a_0(x_j)\, v_k(x_j), \quad \dot{q}_k(0) \approx \frac{\rho L}{np} \sum_{j=1}^{np} b_0(x_j)\, v_k(x_j) \tag{13.26}$$

where np equally-spaced points along the length L, $x_j = j \cdot \Delta x = j \cdot L/np$, are used. For accurate numerical results, np is at least 100.

Window 3.1. Functions free2os and free2os1

MATLAB Functions: **free2os, free2os1**

Purpose:
To plot free response of a second-order distributed dynamic system.

Synopsis:

```
free2os(IC, xr)
[y, tt] = free2os(IC, xr, tf, npts)
free2os1(IC, t)
[y, x] = free2os1(IC, t, npts)
```

Description:
`free2os(IC, xr)` plots the free response of the system at location `xr`. Here `IC` is a 2-by-*np* matrix specifying the initial displacement and velocity $a_0(x)$, $b_0(x)$ at *np* points:

$$\mathtt{IC} = \begin{bmatrix} a_0(x_1) & a_0(x_2) & \cdots & a_0(x_{np}) \\ b_0(x_1) & b_0(x_2) & \cdots & b_0(x_{np}) \end{bmatrix}$$

`[y, tt] = free2os(IC, xr, tf, npts)` returns the computed free response in a vector `y` and times of computation in a vector `tt`, where `tf` is total simulation time and `npts` is the number of time points in the region `[0, tf]`. By default, `tf` = 4*T with T being the first natural period of the system and `npts` = 201.

`free2os1(IC, t)` plots the spatial distribution of the free response at a specific time `t`.

`[y, x] = free2os1(IC, t, npts)` returns the spatial distribution of the free response in a vector `y` and spatial points of computation in a vector `x`, where `npts` is the number of spatial points in the region `[0, L]`. By default, `npts` = 201.

EXAMPLE 3.1

Consider a fixed-free elastic bar of parameters $\rho_v A = 0.3$, $EA = 40$, and $L = 5.0$. Let the bar be subject to the initial disturbances

$$w(x,0) = 0 \text{ for } 0 \le x \le L$$

$$\frac{\partial w(x,0)}{\partial t} = \begin{cases} 0.01x(0.5L - x), & \text{for } 0 \le x \le L/2 \\ 0, & \text{for } L/2 < x \le L \end{cases}$$

See Fig. 13.3.1 for the spatial distribution of the initial velocity. Assume that the free vibration of the bar at its tip ($x = L$) is of interest. It is computed by the following three steps:

(i) Set up the system by

```
>> setbar(0.3, 40, 5, [1 2])
```

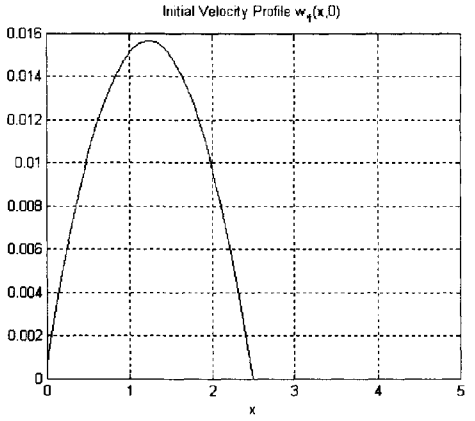


FIGURE 13.3.1 Initial velocity profile.

(ii) Form the initial condition matrix by

```
>> np = 100; L = 5; xx = 0.5*L/np*(1:np);
>> A0 = zeros(1,2*np);
>> B0 = [0.01*xx.*(0.5*L-xx) zeros(1,np)];
>> IC = [A0; B0];
```

(iii) Compute the free response by

```
>> free2os(IC, 5)
```

which yields Fig. 13.3.2.

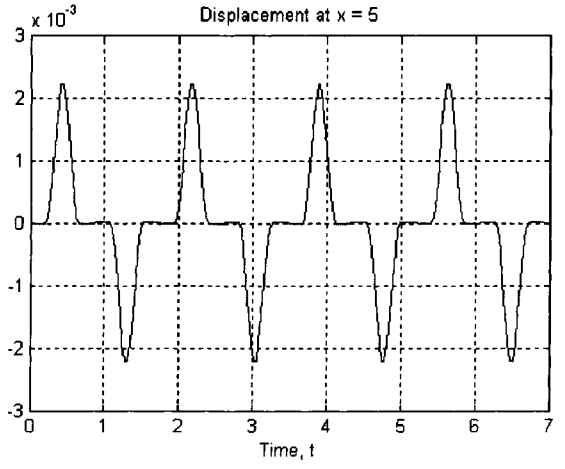


FIGURE 13.3.2

Also, to plot the spatial distribution of the free response at times t = 0, 0.1, 0.2, 0.3, and 0.4 sec., execute the following commands:

```
>> [y1,x] = free2os1(IC, 0);
>> y2 = free2os1(IC, 0.1);
>> y3 = free2os1(IC, 0.2);
>> y4 = free2os1(IC, 0.3);
>> y5 = free2os1(IC, 0.4);
>> plot(x,y1,x,y2,x,y3,x,y4,x,y5)
>> grid; xlabel('x'); ylabel('w(x,t)')
```

which produce Fig. 13.3.3.

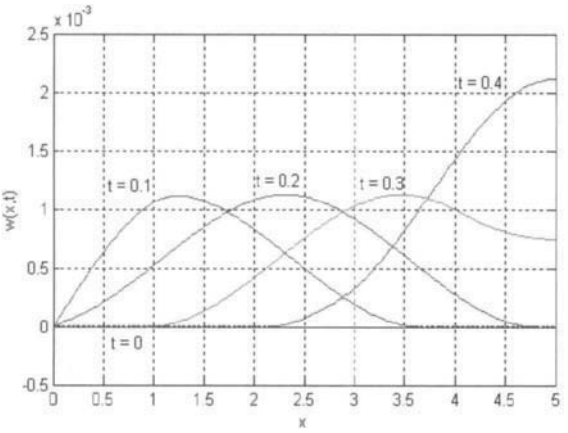


FIGURE 13.3.3

13.3.3 Forced Vibration

For a second-order distributed dynamic system under an external load, its modal coordinates, by Eq. (13.19), are governed by

$$\begin{aligned} \ddot{q}_k(t) + \omega_k^2 q_k(t) &= b_k f(t) \\ q_k(0) = 0, \quad \dot{q}_k(0) &= 0 \end{aligned} \tag{13.27}$$

Here zero initial disturbances have been assumed. The solution of Eq. (13.27) is given in the following two cases:

$$\text{for} \quad \omega_k \neq 0 \text{ (flexible mode)} \quad q_k(t) = \frac{b_k}{\omega_k} \int_0^t \sin \omega_k (t - \tau) f(\tau)\, d\tau \tag{13.28}$$

$$\text{for} \quad \omega_k = 0 \text{ (rigid-body mode)} \quad q_k(t) = b_k \int_0^t (t - \tau) f(\tau)\, d\tau \tag{13.29}$$

Table 13.3.1 lists the modal coordinates of a second-order system under impulsive, step, and sinusoidal loads, where $f(t)$ is from the forcing function $F(x, t) = D(x)f(t)$. The parameter b_k, which is defined in Eq. (13.20), represents the contribution of an external load in the kth

TABLE 13.3.1 Modal Coordinates in Certain Loading Cases

Load	*Modal Coordinates*
Impulse $f(t) = I_0\delta(t)$	Flexible mode ($\omega_k \neq 0$): $q_k(t) = \dfrac{I_0 b_k}{\omega_k}\sin\omega_k t$ Rigid-body mode ($\omega_k = 0$): $q_k(t) = I_0 b_k t$
Step (constant) load $f(t) = F_0$	Flexible mode ($\omega_k \neq 0$): $q_k(t) = \dfrac{F_0 b_k}{\omega_k^2}(1 - \cos\omega_k t)$ Rigid-body mode ($\omega_k = 0$): $q_k(t) = \dfrac{1}{2}F_0 b_k t^2$
Sinusoidal load $f(t) = A\sin(\omega t + \phi_0)$	Flexible mode ($\omega_k \neq 0$): Case 1. Nonresonant vibration ($\omega \neq \omega_k$) $q_k(t) = Ab_k H_k \sin(\omega t + \phi_0) - Ab_k H_k(\sin\phi_0\cos\omega_k t + \gamma_k \sin\omega_k t)$ where $H_k = \dfrac{1}{\omega_k^2 - \omega^2}$, $\gamma_k = \omega\cos\phi_0/\omega_k$ Case 2. Resonant vibration ($\omega = \omega_k$) $q_k(t) = -\dfrac{Ab_k}{2\omega_k}t\cos(\omega_k t + \phi_0) + \dfrac{Ab_k}{2\omega_k^2}\cos(\phi_0)\sin\omega_k t$ Rigid-body mode ($\omega_k = 0$): $q_k(t) = \dfrac{Ab_k}{\omega^2}[\sin\phi_0 - \sin(\omega t + \phi_0)] + \dfrac{Ab_k}{\omega}t\cdot\cos\phi_0$

mode. For a pointwise load at $x = x_f$, $F(x,t) = \delta(x - x_f)f(t)$, where δ is the delta function (see Fig. 13.3.4a),

$$b_k = v_k(x_f) \tag{13.30}$$

For a force uniformly distributed in the region $x_1 \leq x \leq x_2$, $F(x,t) = (h(x - x_1) - h(x - x_2))f(t)$ where $h(x)$ is the unite step function (see Fig. 13.3.4b),

$$b_k = \int_{x_1}^{x_2} v_k(x)\,dx \tag{13.31}$$

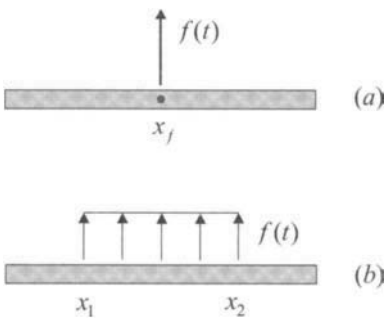


FIGURE 13.3.4 Two types of force distribution.

This Toolbox provides functions `forced2os` and `forced2os1` for computing forced vibration of a second-order distributed dynamic system; see Window 3.2.

Window 3.2. Functions forced2os and forced2os1

MATLAB Functions: **forced2os, forced2os1**

Purpose:
To plot the forced response of a second-order distributed dynamic system.

Synopsis:

```
forced2os(Load_Spec, xr)
[y, tt] = forced2os(Load_Spec, xr, tf, npts)
forced2os1(Load_Spec, t)
[y, x] = forced2os1(Load_Spec, t, npts)
```

Description:
`forced2os(Load_Spec, xr)` plots the forced response of the system at location `xr`. Here, `Load_Spec` is a vector specifying an external load according to Table 13.3.2.

`[y, tt] = forced2os(Load_Spec, xr, tf, npts)` returns computed forced response in a vector `y` and times of computation in a vector `tt`, where `tf` is total simulation time and `npts` is the number of time points in the region `[0, tf]`. By default, `tf` = 4*T with T being the first natural period of the system and `npts` = 201.

`forced2os1(Load_Spec, t)` plots the spatial distribution of the forced response at a specific time `t`. Refer to Table 13.3.2 for assignment of `Load_Spec`.

`[y, x] = forced2os1(Load_Spec, t, npts)` returns the spatial distribution of the forced response in a vector `y` and spatial points of computation in a vector `x`, where `npts` is the number of spatial points in the region `[0, L]` for simulation. By default, `npts` = 201.

TABLE 13.3.2 External Loads and Load Specification Vector `Load_Spec`

External Load	*F(x,t)*	`Load_Spec`
(L0) Pointwise impulse load	$I_0\delta(t)\delta(x-x_f)$	[0 x_f I_0]
(L1) Pointwise step load	$F_0\delta(x-x_f)$	[1 x_f F_0]
(L2) Pointwise sinusoidal load	$A\sin(\omega t+\phi_0)\delta(x-x_f)$	[2 x_f A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
(L10) Uniformly distributed impulse load	$I_0\delta(t)$, for $x_1 \le x \le x_1$	[10 x_1 x_2 I_0]
(L11) Uniformly distributed step load	F_0, for $x_1 \le x \le x_1$	[11 x_1 x_2 F_0]
(L12) Uniformly distributed sinusoidal load	$A\sin(\omega t+\phi_0)$, for $x_1 \le x \le x_1$	[12 x_1 x_2 A ω ϕ_0] ω in rad/s, ϕ_0 in degrees

EXAMPLE 3.2

A taut string of parameters $\mu = 0.3$, $T = 40$, $L = 5$ is fixed at the left end and supported by a spring ($k_s = 5$) at the right end. A pointwise sinusoidal force $f(t) = 3\cos(11.2t)$ is applied to the string at the midpoint $x_f = 2.5$. The forced displacement of the string at its right end ($x = 5$) is computed and plotted in Fig. 13.3.5 by

```
>> setstring(0.3, 40, 5, [1 3 5])
>> load = [2 2.5 3 11.2 90]; xr = 5;
>> forced2os(load, xr)
```

The time response in the figure indicates resonant vibration. This is because the excitation frequency, $\omega = 11.2$, is close to the second natural frequency of the string, $\omega_2 = 11.1793$. (The natural frequencies of the string are computed by function `setstring`.)

Now consider a uniformly distributed step load (L11 in Table 13.3.2)

$$F(x,t) = \begin{cases} -1 & \text{for } 1 \le x \le 4 \\ 0, & \text{otherwise} \end{cases}$$

Under this load, the spatial distributions of the string response at times t = 0, 0.05, 0.1, 0.15, and 0.2 are plotted in Fig. 13.3.6 by the following commands:

```
>> [y1, x] = forced2os1([11 1 4 -1], 0);
>> y2 = forced2os1([11 1 4 -1], 0.05);
>> y3 = forced2os1([11 1 4 -1], 0.1);
>> y4 = forced2os1([11 1 4 -1], 0.15);
>> y5 = forced2os1([11 1 4 -1], 0.2);
>> plot(x,y1,x,y2,x,y3,x,y4,x,y5)
>> grid; xlabel('x'); ylabel('w(x,t)')
```

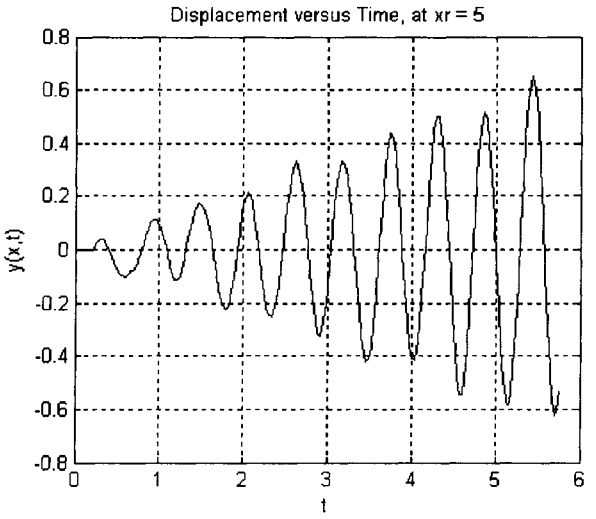


FIGURE 13.3.5

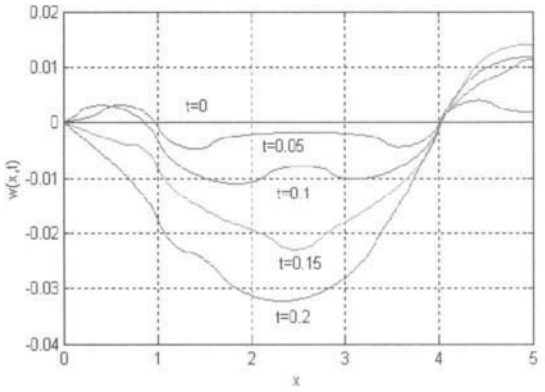


FIGURE 13.3.6

13.3.4 Total Response

According to the previous sections, the total response of a second-order distributed dynamic system subject to both initial and external disturbances is

$$w(x,t) = \sum_{k=1}^{\infty} v_k(x) q_k(t) \tag{13.32}$$

where the modal coordinates are given by

$$q_k(t) = q_k(0)\cos\omega_k t + \dot{q}_k(0)\frac{1}{\omega_k}\sin\omega_k t + \frac{b_k}{\omega_k}\int_0^t \sin\omega_k(t-\tau)f(\tau)\,d\tau \tag{13.33}$$

In the case of rigid-body mode ($\omega_k \to 0$), Eq. (13.33) reduces to

$$q_k(t) = q_k(0) + \dot{q}_k(0)\,t + b_k\int_0^t (t-\tau)f(\tau)\,d\tau$$

Equation (13.32) can be rewritten as

$$w(x,t) = w_I(x,t) + w_F(x,t) \tag{13.34}$$

with

$$w_I(x,t) = \sum_{k=1}^{\infty} v_k(x)\left(q_k(0)\cos\omega_k t + \dot{q}_k(0)\frac{1}{\omega_k}\sin\omega_k t\right) \tag{13.35a}$$

$$w_F(x,t) = \sum_{k=1}^{\infty} v_k(x)\frac{b_k}{\omega_k}\int_0^t \sin\omega_k(t-\tau)f(\tau)\,d\tau \tag{13.35b}$$

The $w_I(x,t)$ is the free response of the system due to the initial disturbances $w(x,0) = a_0(x)$ and $\frac{\partial}{\partial t}w(x,0) = b_0(x)$, which can be obtained by functions `free2os` and `free2os1`; see Window 3.1. The $w_F(x,t)$ is the forced response of the system subject to an external load

$F(x,t) = D(x)f(t)$, which can be determined by functions `forced2os` and `forced2os1`; see Window 3.2. So, the total dynamic response of a second-order distributed system under both initial and external disturbances can be obtained by superposition; see Window 3.3.

Window 3.3. Determination of total response

Disturbances
Initial disturbances $w(x,0) = a_0(x)$ and $\frac{\partial}{\partial t}w(x,0) = b_0(x)$, specified by matrix `IC`
External loads $F(x,t) = D(x)f(t)$, specified by vector `Load_Spec`

Total response
To determine time response at a location `xr`, execute the commands:

```
[y0, tt] = free2os(IC, xr);
yf = forced2os(Load_Spec, xr);
plot(tt, y0+yf)
```

To determine spatial distribution of the system response at a specific time `t`, execute the commands:

```
[y0, x] = free2os1(IC, t);
yf = forced2os1(Load_Spec, t);
plot(x, y0+yf)
```

Here `plot` is a standard MATLAB function.

If a system is under multiple external loads, say $F_1(x,t)$, $F_2(x,t),\ldots,F_m(x,t)$, its total response can be obtained by

```
y_total = free2os(IC, xr) + forced2os(Load_1, xr) + forced2os(Load_2, xr)
        + ··· + forced2os(Load_m, xr);
```

or

```
y_total = free2os1(IC, t) + forced2os1(Load_1, t) + forced2os(Load_2, t)
        + ··· + forced2os(Load_m, t);
```

where `Load_1`, `Load_2`, ..., `Load_m` are vectors that specify the external forces by Table 13.3.2.

EXAMPLE 3.3

In Fig. 13.3.7, a circular shaft of parameters $\rho_v J = 1$, $GJ = 100$, and $L = 3$ is constrained by a spring ($k = 20$) at its left end and fixed at its right end. The shaft has the initial conditions

$$\theta(x,0) = 0.01(L-x)^2, \quad \frac{\partial}{\partial t}\theta(x,0) = 0$$

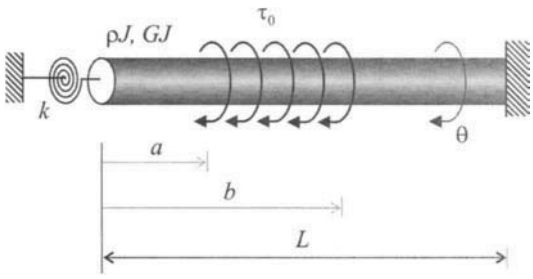


FIGURE 13.3.7

and is subject to a uniformly distributed step torque $\tau(x,t) = \tau_0$ in the region $a \le x \le b$, with $\tau_0 = 10$, $a = 1$, and $b = 2$. Under the disturbances, the spatial distribution of the response of the shaft at times $t = 0$, 0.05, 0.1, 0.15, and 0.2 is plotted in Fig. 13.3.8 by the following commands

```
>> setshaft(1, 100, 3, [3 20 1])
>> np = 30000; xx = 3/np*(1:np);
>> IC = [0.01*(3-xx).^2; zeros(1,np)];   % initial disturbance
>> load = [11 1 2 10];                   % external torque
>> [y0, x] = free2os1(IC, 0);
>> y1 = y0 + forced2os1(load, 0);
>> y2 = free2os1(IC, 0.01) + forced2os1(load, 0.05);
>> y3 = free2os1(IC, 0.02) + forced2os1(load, 0.1);
>> y4 = free2os1(IC, 0.03) + forced2os1(load, 0.15);
>> y5 = free2os1(IC, 0.04) + forced2os1(load, 0.2);
>> plot(x,y1,x,y2,x,y3,x,y4,x,y5);
>> grid; xlabel('x');  ylabel('\theta(x,t)')
```

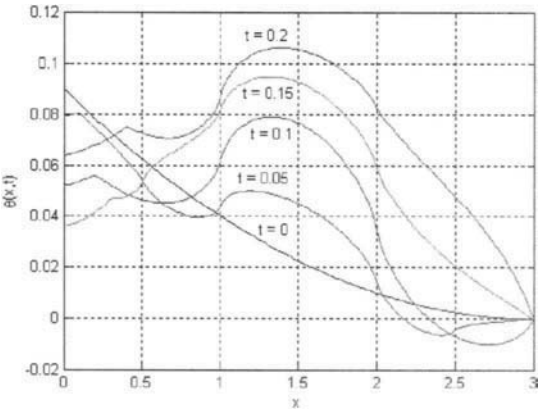


FIGURE 13.3.8

Also, the time history of the shaft at the left end ($x = 0$) is plotted in Fig. 13.3.9 by

```
>> [y0, tt] = free2os(IC, 0);
>> y = y0 + forced2os(load, 0);
>> plot(tt,y); grid; xlabel('t'); ylabel('\theta(x=0,t)')
```

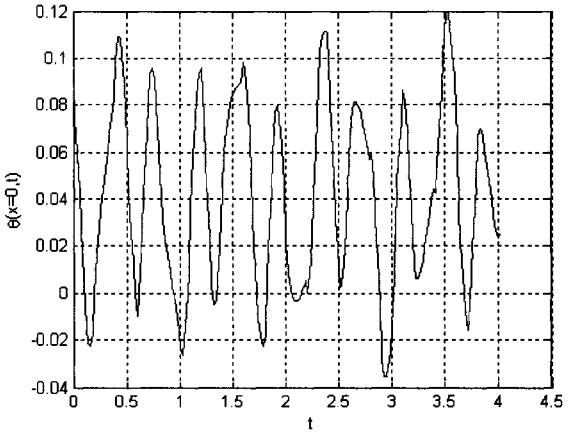


FIGURE 13.3.9

13.3.5 Animation of Time Response

The time response obtained in Sections 13.3.2 to 13.3.4 can be animated by function `movie2os` from the Toolbox; see Window 3.4.

Window 3.4. Function `movie2os`

MATLAB Function: **`movie2os`**

Purpose:
To animate the time response of a second-order distributed dynamic system in its spatial region.

Synopsis:

```
movie2os(Load_Spec, IC)
movie2os(Load_Spec, IC, IS_control, n_frames, nfpp, frame_bounds)
F = movie2os(Load_Spec, IC, IS_control, n_frames, nfpp,
    frame_bounds)
```

Description:
`movie2os(Load_Spec, IC)` animates the time response of the system subject to an external load that is specified by a vector `Load_Spec` according to Table 13.3.2, and initial disturbances that are specified by matrix `IC` according to Window 3.1. `Load_Spec` or `IC` can be replaced by `[ ]` or 0 if no external force or initial disturbances exist. Thus, `movie2os(Load_Spec)` animates the forced response, and `movie2os(0, IC)` does the free response of the system.

Window 3.4. Function `movie2os` (*Continued*)

`movie2os(Load_Spec, IC, IS_control, n_frames, nfpp, frame_bounds)` plays the animated vibration in a specified manner, where `IS_control` is an animation control parameter with the options

```
IS_control = 0    play frames continuously
IS_control = 1    play frames one by one
```

`n_frames` is the total number of frames, `nfpp` is the number of frames per first natural period of the system ($T_1 = 2\pi/\omega_1$), and the vector `frame_bounds = [xmin xmax ymin ymax]` sets up the bounds of frames. By default, `IS_control = 0`, `n_frames = 97`, `nfpp = 12`, and `frame_bounds` is automatically set up. Any of `IS_control`, `n_frames`, `nfpp`, `frame_bounds` can be replaced by `[ ]` as a placeholder for its default value.

`F = movie2os(Load_Spec, IC, IS_control, n_frames, nfpp, frame_bounds)` returns a set of frames of the time response in a matrix `F`, which can be played back by the MATLAB function `movie(F)`.

EXAMPLE 3.4

Consider a fixed-free elastic bar of parameters $\rho_v A = 0.3$, $EA = 40$, and $L = 5.0$. Let the bar be subject to a uniformly distributed step load

$$F(x,t) = \begin{cases} -1.5, & \text{for } 3.3 \le x \le 3.7 \\ 0, & \text{otherwise} \end{cases}$$

The forced vibration of the bar is animated by the commands

```
>> setbar(0.3, 40, 5, [1 1])
>> movie2os([11 3.3 3.7 -1.5])
```

Shown in Fig. 13.3.10 are six frames of the animated bar displacement at time $t_i = i \times h$, $i = 0, 2, \ldots, 5$, where $h = \pi/(6\omega_1) = 0.07169$ with $\omega_1 = 7.2552$ being the first natural frequency of the bar.

13.3.6 Frequency Response

Consider a second-order distributed system subject to a pointwise harmonic (sinusoidal) load at $x = x_f$

$$\rho \frac{\partial^2}{\partial t^2} w(x,t) - \alpha_{ST} \frac{\partial^2}{\partial x^2} w(x,t) = f_0 e^{j\omega t} \delta(x - x_f), \quad 0 \le x \le L \tag{13.36}$$

where f_0 and ω are the amplitude and frequency of the load, $\delta(\cdot)$ is the delta function, and $j = \sqrt{-1}$. The steady-state displacement of the system can be written as

$$w_{ss}(x,t) = W(x) e^{j\omega t} \tag{13.37}$$

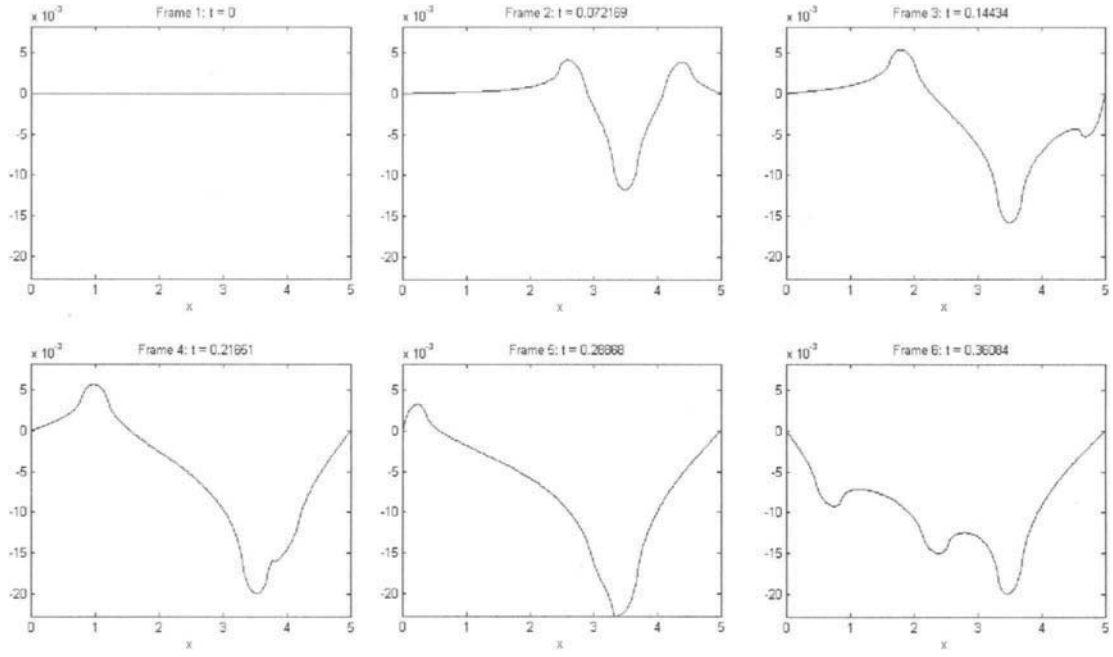


FIGURE 13.3.10 Frames of the animated vibration in Example 3.4.

Substitute Eq. (13.37) into (13.36), to obtain

$$-\rho\omega^2 W(x) - \alpha_{ST}\frac{d^2}{dx^2}W(x) = f_0\,\delta(x - x_f), \quad 0 \le x \le L \tag{13.38}$$

The boundary conditions of $W(x)$ are the same as those given in Eqs. (13.3). The $W(x)$ is called the *frequency response* of the system. In practice, the magnitude and phase of the frequency response

$$H = |W(x,\omega)|, \quad \phi = \angle W(x,\omega) \tag{13.39}$$

are plotted against the excitation frequency ω.

The frequency response can be determined by the following two methods.

Method 1. Modal Expansion Following Section 13.3.1, express $W(x)$ by the series

$$W(x) = \sum_{k=1}^{\infty} Q_k v_k(x) \tag{13.40}$$

where $v_k(x)$ are the system eigenfunctions, and Q_k are modal coefficients. Substituting Eq. (13.40) into Eq. (13.38), and applying the orthonormal relations (13.7), yields

$$-\omega^2 Q_k + \omega_k^2 Q_k = f_0\, v_k(x_f) \tag{13.41}$$

It follows that

$$Q_k = \frac{f_0 v_k(x_f)}{\omega_k^2 - \omega^2} \tag{13.42}$$

Thus, the frequency response is

$$W(x) = f_0 \sum_{k=1}^{\infty} \frac{v_k(x_f)}{\omega_k^2 - \omega^2} v_k(x) \tag{13.43}$$

Method 2. Distributed Transfer Function Formulation The governing equation (13.38) is cast into an equivalent state-space form

$$\frac{d}{dx}\{\eta(x)\} = [F(\omega)]\{\eta(x)\} - \frac{1}{\alpha_{ST}} f_0 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \delta(x - x_f), \quad 0 \le x \le L \tag{13.44}$$

where

$$\{\eta(x)\} = \begin{pmatrix} W(x) \\ \dfrac{d}{dx} W(x) \end{pmatrix}, \quad [F(\omega)] = \begin{bmatrix} 0 & 1 \\ -\beta^2 & 0 \end{bmatrix} \tag{13.45}$$

with $\beta = \omega\sqrt{\rho/\alpha_{ST}}$. The system boundary conditions can also be written as

$$[M_b]\{\eta(0)\} + [N_b]\{\eta(L)\} = 0 \tag{13.46}$$

where $[M_b]$ and $[N_b]$ are 2-by-2 matrices specifying the boundary conditions at the left and right ends, respectively. For instance, for a shaft with its left end fixed and right end supported by a spring of coefficient k_s, the boundary condition matrices are

$$[M_b] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad [N_b] = \begin{bmatrix} 0 & 0 \\ k & GJ \end{bmatrix}$$

By the Distributed Transfer Function Method (see Appendix C), the solution of Eq. (13.44) subject to the boundary condition (13.46) is obtained as

$$\eta(x) = -\frac{f_0}{\alpha_{ST}} [G(x, x_f)] \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{13.47}$$

where

$$[G(x,\xi)] = \begin{cases} [H(x)][M_b]\, e^{-[F(\omega)]\xi}, & x \ge \xi \\ -[H(x)][N_b]\, e^{[F(\omega)](L-\xi)}, & x \le \xi \end{cases} \tag{13.48a}$$

with

$$[H(x)] = e^{[F(\omega)]x} \left([M_b] + [N_b]\, e^{[F(\omega)]L} \right)^{-1} \tag{13.48b}$$

The $e^{[F(\omega)]x}$ is an exponential matrix of the form

$$e^{[F(\omega)]x} = \begin{bmatrix} \cos \beta x & \frac{1}{\beta} \sin \beta x \\ -\beta \sin \beta x & \cos \beta x \end{bmatrix} \tag{13.49}$$

Write

$$[G(x, \xi)] = \begin{bmatrix} g_{11}(x, \xi) & g_{12}(x, \xi) \\ g_{21}(x, \xi) & g_{22}(x, \xi) \end{bmatrix} \tag{13.50}$$

By Eq. (13.47), the frequency response of the system is given in the exact and closed form

$$W(x) = -\frac{f_0}{\alpha_{ST}} g_{12}(x, \xi) \tag{13.51}$$

This Toolbox provides a function `frf2os` for plotting the frequency response of a second-order distributed system subject to a pointwise sinusoidal load; see Window 3.5.

Window 3.5. Function `frf2os`

MATLAB Function: **`frf2os`**

Purpose:
To plot the magnitude and phase of the frequency response $W(x)$ of a second-order distributed dynamic system by Eq. (13.51).

Synopsis:

```
frf2os(xr, xf, f0)
frf2os(xr, xf)
[y, freq] = frf2os(xr, xf, f0, omg0, omgf, npts)
```

Description:
`frf2os(xr, xf, f0)` plots the magnitude and phase of the system frequency response at position `xr` versus frequency. The system is under a harmonic load of amplitude `f0`, applied at position `xf`.

`frf2os(xr, xf)` plots the frequency response for `f0 = 1`.

`[y, freq] = frf2os(xr, xf, f0, omg0, omgf, npts)` returns the frequency response in a two-row matrix `y` and frequencies of simulation in a vector `freq`. Here, `omg0` and `omgf` are the lower and upper bounds of the frequency region, and `npts` is the number of frequency points in the region `[omg0, omgf]` for simulation. By default, `omg0 = 0`, `omgf` is automatically adjusted such that it is at least larger than the third natural frequency, and `npts = 1,001`. With `y` and `freq`, the frequency response can be plotted as follows:

```
plot(freq, y(1,: ))   for magnitude
plot(freq, y(2,: ))   for phase
```

EXAMPLE 3.5

A taut string of linear density $\mu = 0.5$, tension $T = 8$, and length $L = 1$ is fixed at the left end, and supported at the right end by a spring of coefficient $k_s = 10$. A sinusoidal force of unit amplitude is applied to the string at $x_f = 0.8$. The amplitude and phase of the steady-state displacement of the string at $x = 0.3$ is plotted against the excitation frequency in Fig. 13.3.11 by the following commands

```
>> setstring(0.5, 8, 1, [1 3 10])
>> frf2os(0.3, 0.8)
```

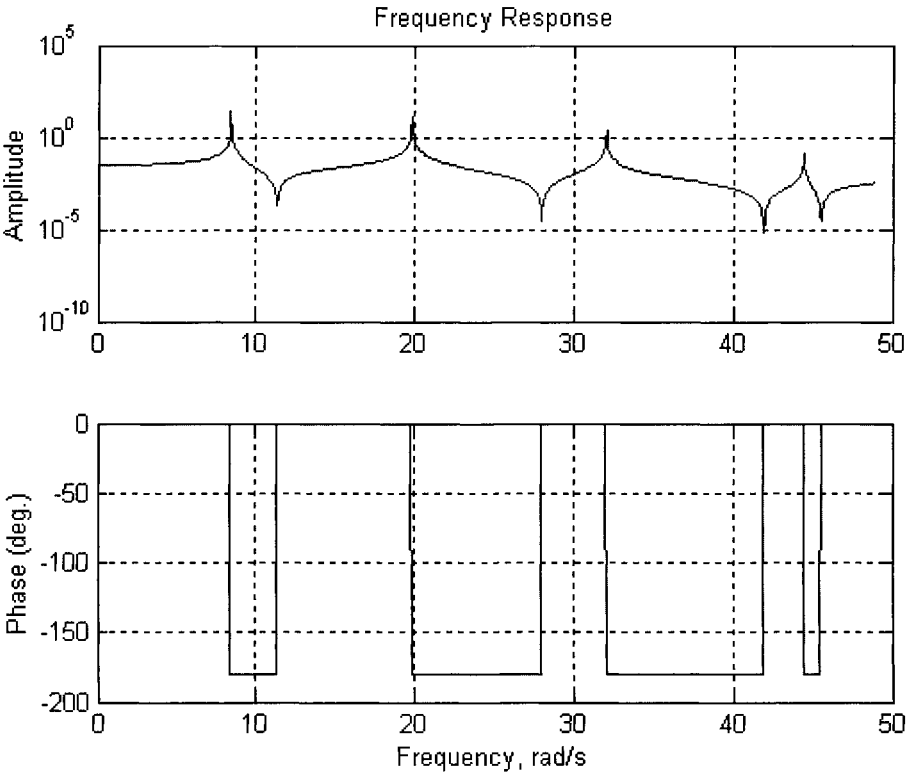


FIGURE 13.3.11

13.4 Free Vibration of Stepped Systems

This section is concerned with free vibration of stepped bars, shafts, and strings that are constrained by springs and combined with lumped masses. The description of such structures is given in Section 13.4.1, two methods of free vibration analysis are introduced in Section 13.4.2, and related MATLAB functions are presented in Section 13.4.3.

13.4.1 System Description

Stepped Bars in Longitudinal Vibration The schematic of a stepped elastic bar in longitudinal vibration is shown in Fig. 13.4.1, where the structure is an assembly of N uniform bar

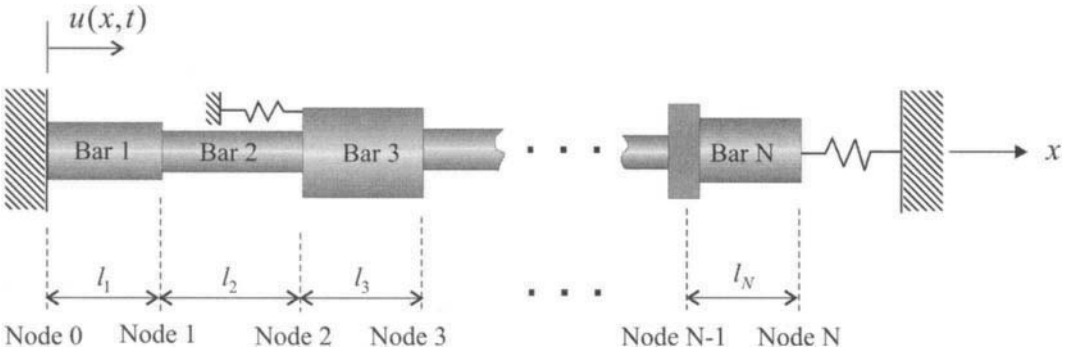


FIGURE 13.4.1 Schematic of a stepped bar.

elements that are serially connected at N-1 points called *nodes*. At those nodes, there may be spring constrains and mounted lumped masses (see nodes 2 and N-1 for instance). The left and right ends of the structure are labeled as Nodes 0 and N.

The longitudinal displacement $u_i(x,t)$ of the ith element is governed by the differential equation

$$\rho_v A_i \frac{\partial^2}{\partial t^2} u_i(x,t) - EA_i \frac{\partial^2}{\partial x^2} u_i(x,t) = 0, \quad 0 \le x \le l_i \tag{13.52}$$

where $\rho_v A_i$, EA_i, and l_i are the linear density, longitudinal rigidity, and length of the element, respectively, and x is a local coordinate of the element. The connection of the ith and $(i+1)$th bar elements at node i satisfies the following compatibility conditions

$$\begin{gathered} u_i(l_i,t) = u_{i+1}(0,t) \\ m_c \frac{\partial^2}{\partial t^2} u_i(l_i,t) + EA_i \frac{\partial}{\partial x} u_i(l_i,t) + k_c u_i(l_i,t) = EA_{i+1} \frac{\partial}{\partial x} u_{i+1}(0,t) \end{gathered} \tag{13.53}$$

where k_c is the coefficient of a spring that is imposed to the node as a constraint, and m_c is a lumped mass that is mounted at the node.

Stepped Shafts in Torsional Vibration A stepped torsional shaft is an assembly of N uniform shaft elements connected at N-1 nodes; see Fig 13.4.2. At these nodes, there may be torsional

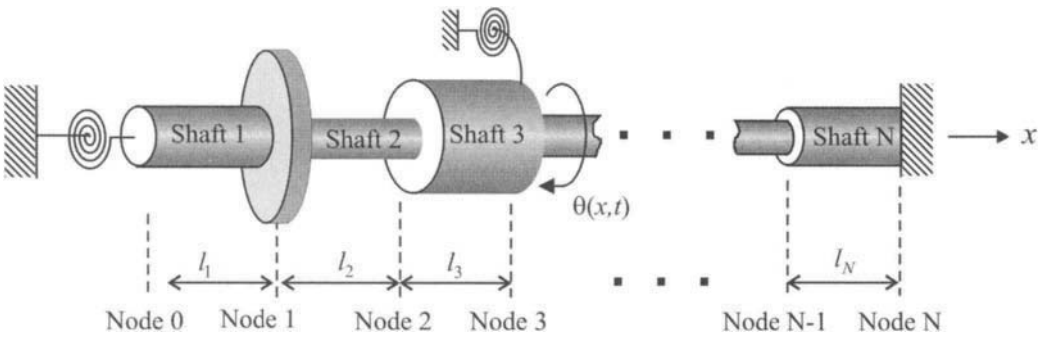


FIGURE 13.4.2 Schematic of a stepped shaft.

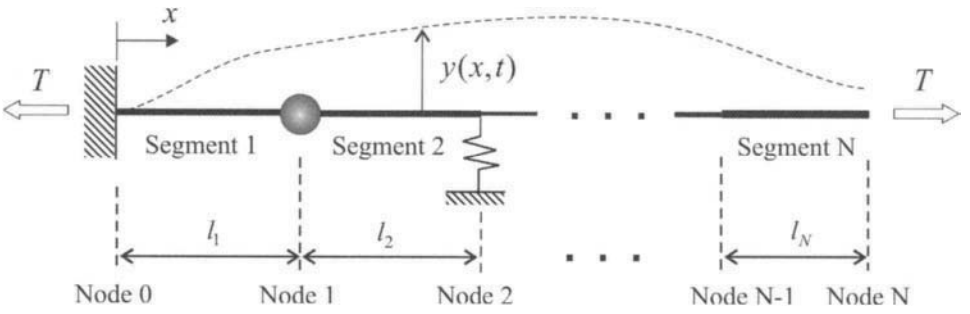


FIGURE 13.4.3 Schematic of a stepped taut string.

spring constrains and mounted rigid disks (see nodes 1 and 3 for instance). The rotational displacement $\theta_i(x,t)$ of the ith element is governed by

$$\rho_v J_i \frac{\partial^2}{\partial t^2}\theta_i(x,t) - GJ_i \frac{\partial^2}{\partial x^2}\theta_i(x,t) = 0, \quad 0 \le x \le l_i \tag{13.54}$$

where $\rho_v J_i$, GJ_i, and l_i are the linear density, torsional rigidity, and length of the element, respectively, and x is a local coordinate of the element. The connection of the ith and $(i+1)$th shaft elements at node i is described by the following compatibility equations

$$\begin{gathered} \theta_i(l_i,t) = \theta_{i+1}(0) \\ I_D \frac{\partial^2}{\partial t^2}\theta_i(l_i,t) + GJ_i \frac{\partial}{\partial x}\theta_i(l_i,t) + k_\tau \theta_i(l_i,t) = GJ_{i+1}\frac{\partial}{\partial x}\theta_{i+1}(0,t) \end{gathered} \tag{13.55}$$

where k_τ is the coefficient of a torsional spring that is imposed to the node as a constraint, and I_D is the moment of inertia of the rigid disk mounted at the node.

Stepped Taut Strings in Lateral Vibration A stepped taut string is an assembly of N uniform string segments connected at N-1 nodes; see Fig. 13.4.3. At these nodes, there may be spring constrains and mounted lumped masses (see nodes 1 and 2 for instance). The lateral displacement $y_i(x,t)$ of the ith segment is governed by

$$\mu_i \frac{\partial^2}{\partial t^2} y_i(x,t) - T\frac{\partial^2}{\partial x^2} y_i(x,t) = 0, \quad 0 \le x \le l_i \tag{13.56}$$

where μ_i and l_i are the linear density and length of the element, respectively, and x is a local coordinate of the segment. The string tension T is the same for every segment, while the linear density may vary from segment to segment. The connection of the ith and $(i+1)$th string segments at node i is characterized by the compatibility conditions

$$\begin{gathered} y_i(l_i,t) = y_{i+1}(0) \\ m_c \frac{\partial^2}{\partial t^2} y_i(l_i,t) + T\frac{\partial}{\partial x} y_i(l_i,t) + k_c y_i(l_i,t) = T\frac{\partial}{\partial x} y_{i+1}(0,t) \end{gathered} \tag{13.57}$$

where k_c is the coefficient of a translational spring that is imposed to the node as a constraint, and m_c is a lumped mass mounted at the node.

Unified Governing Equations The governing equations of the aforementioned stepped systems can be written in the following unified form:

Equation of motion of the ith element:

$$\rho_i \frac{\partial^2}{\partial t^2} w_i(x,t) - \alpha_i^{ST} \frac{\partial^2}{\partial x^2} w_i(x,t) = 0, \quad 0 \le x \le l_i \tag{13.58}$$

Compatibility conditions at node i:

$$\begin{gathered} w_i(l_i,t) = w_{i+1}(0,t) \\ m_i \frac{\partial^2}{\partial t^2} w_i(l_i,t) + \alpha_i^{ST} \frac{\partial}{\partial x} w_i(l_i,t) + k_i w_i(l_i,t) = \alpha_{i+1}^{ST} \frac{\partial}{\partial x} w_{i+1}(0,t) \end{gathered} \tag{13.59}$$

where $w_i(x,t)$, ρ_i, and α_i^{ST} are the displacement, inertia, and stiffness (rigidity) of the ith element, respectively; k_i and m_i are the spring coefficient and lumped mass that are imposed at node i as a constraint.

The boundary conditions of such a stepped structural system at node 0 ($x = 0$) and node N ($x = L$) are given in Table 13.4.1, where L is the total length of the stepped system. Boundary conditions (B1) to (B3) are the same as those listed in Table 13.2.5 for a single-body system. For boundary conditions (B3) and (B4), k_0 and k_N are coefficients of end springs, and m_0 and m_N are inertia parameters of end masses or end disks.

TABLE 13.4.1 Specification of Boundary Conditions of Stepped Systems

Boundary Conditions	**BC_Spec = [BC_Left BC_Right]**	
	Left end (x = 0) BC_Left	*Right end (x = L)* BC_Right
(B1) Fixed end		
At $x = 0$: $w_1(0,t) = 0$	[1]	[1]
At $x = L$: $w_N(l_N,t) = 0$		
(B2) Free end		
At $x = 0$: $\dfrac{\partial w_1(0,t)}{\partial x} = 0$	[2]	[2]
At $x = L$: $\dfrac{\partial w_N(l_N,t)}{\partial x} = 0$		
(B3) End spring		
At $x = 0$: $k_0 w_1(0,t) - \alpha_1^{ST} \dfrac{\partial w_1(0,t)}{\partial x} = 0$	[3 k_0]	[3 k_N]
At $x = L$: $k_N w_N(l_N,t) + \alpha_N^{ST} \dfrac{\partial w_N(l_N,t)}{\partial x} = 0$		
(B4) End spring-mass		
At $x = 0$: $m_0 \dfrac{\partial^2 w_1(0,t)}{\partial t^2} + k_0 w_1(0,t) - \alpha_1^{ST} \dfrac{\partial w_1(0,t)}{\partial x} = 0$	[4 k_0 m_0]	[4 k_N m_N]
At $x = L$: $m_N \dfrac{\partial^2 w_N(l_N,t)}{\partial t^2} + k_N w_N(l_N,t) + \alpha_N^{ST} \dfrac{\partial w_N(l_N,t)}{\partial x} = 0$		

13.4.2 Free Vibration Analysis

In the free vibration analysis of a stepped system, the displacement of the ith element is written in the form

$$w_i(x,t) = v_i(x)e^{j\omega t}, \quad j = \sqrt{-1} \tag{13.60}$$

Substituting Eq. (13.60) into Eqs. (13.58) and (13.59) yields

$$\omega^2 v_i(x) + c_i^2 \frac{d^2}{dx^2} v(x) = 0, \quad 0 \le x \le l_i \tag{13.61}$$

and

$$\begin{gathered} v_i(l_i) = v_{i+1}(0) \\ -\omega^2 m_i v_i(l_i) + \alpha_i^{ST} \frac{d}{dx} v_i(l_i) + k_i v_i(l_i) = \alpha_{i+1}^{ST} \frac{d}{dx} v_{i+1}(0) \end{gathered} \tag{13.62}$$

where $c_i = \sqrt{\alpha_i^{ST}/\rho_i}$. Equations (13.61) and (13.62), along with appropriate boundary conditions, define an eigenvalue problem, for which ω is a natural frequency of the stepped system and $v_i(x)$ describes the associate mode shape of the ith element. The natural frequencies can be arranged in an ascending order

$$0 \le \omega_1 < \omega_2 < \omega_3 < \cdots \tag{13.63}$$

The eigenfunctions can be normalized such that

$$\sum_{i=1}^{N} \rho_i \int_0^{l_i} v_i^2(x)dx + m_0 v_1^2(0) + m_1 v_1^2(l_1) + \cdots + m_N v_N^2(l_N) = 1 \tag{13.64}$$

Two analytical methods for the above eigenvalue problem are introduced: the classical boundary value approach and the Distributed Transfer Function Method.

Method 1. Boundary Value Approach To solve the above eigenvalue problem, assume the solution of Eq. (13.61) as

$$v_i(x) = a_i \cos \beta_i x + b_i \sin \beta_i x, \quad i = 1, 2, \ldots, \mathrm{N} \tag{13.65}$$

where

$$\beta_i = \frac{\omega}{c_i} = \omega \sqrt{\frac{\rho_i}{\alpha_i^{ST}}} \tag{13.66}$$

and a_1 and a_2 are constants to be determined. Substitute Eq. (13.65) into the compatibility condition (13.62) and the boundary conditions of the system, to obtain the homogeneous equations

$$[D(\omega)]\{B\} = 0 \tag{13.67}$$

where $[D(\omega)]$ is a 2N-by-2N matrix with elements as transcendental functions of ω and $\{B\}$ is a 2N vector of the form

$$\{B\} = \begin{pmatrix} a_1 & b_1 & a_2 & b_2 & \cdots & a_N & b_N \end{pmatrix}^T \tag{13.68}$$

The characteristic equation of the stepped system is

$$\Delta(\omega) = \det [D(\omega)] = 0 \tag{13.69}$$

the roots of which are the natural frequencies of the stepped system. For a known natural frequency ω_k, a nonzero vector $\{B\}$ is obtained from $[A(\omega_k)]\{B\} = 0$, which in turn gives the associate modal shape by Eq. (13.65).

Method 2. The Distributed Transfer Function Method The eigenvalue problem of a stepped system can be conveniently solved by the Distributed Transfer Function Method (DTFM), which is presented in Appendix C. In the DTFM, Eq. (13.61) is cast into an equivalent state-space form

$$\frac{d}{dx}\{\eta_i(x)\} = [F_i]\,\eta_i(x), \quad 0 \le x \le l_i \tag{13.70}$$

where

$$\{\eta_i(x)\} = \begin{pmatrix} v_i(x) \\ \dfrac{d}{dx} v_i(x) \end{pmatrix}, \quad [F_i] = \begin{bmatrix} 0 & 1 \\ -\beta_i^2 & 0 \end{bmatrix} \tag{13.71}$$

with β_i being given in Eq. (13.66). With $\{\eta_i(x)\}$, the compatibility conditions (13.62) are written as

$$\{\eta_{i+1}(0)\} = [T_i]\{\eta_i(l_i)\} \tag{13.72}$$

where

$$[T_i] = \begin{bmatrix} 1 & 0 \\ \dfrac{k_i - \omega^2 m_i}{\alpha_{i+1}^{ST}} & \dfrac{\alpha_i^{ST}}{\alpha_{i+1}^{ST}} \end{bmatrix}. \tag{13.73}$$

The boundary conditions at nodes 0 and N can also be written in the state-space form:

$$[M_b]\{\eta_1(0)\} + [N_b]\{\eta_N(l_n)\} = 0 \tag{13.74}$$

where $[M_b]$ and $[N_b]$ are 2-by-2 boundary condition matrices. Table 13.4.2 lists the boundary condition matrices for the four types of boundary conditions given in Table 13.4.1.

The solution of Eq. (13.70) can be expressed by

$$\{\eta_i(x)\} = e^{[F_i]x}\{\eta_i(0)\}, \quad 0 \le x \le l_i \tag{13.75}$$

where the exponential matrix is given by

$$e^{[F_i]x} = \begin{bmatrix} \cos\beta_i x & \dfrac{1}{\beta_i}\sin\beta_i x \\ -\beta_i \sin\beta_i x & \cos\beta_i x \end{bmatrix} \tag{13.76}$$

TABLE 13.4.2 Boundary Matrices

Boundary Conditions	Left End ($x = 0$) $[M_b]$	Right End ($x = L$) $[N_b]$
(B1) Fixed end	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$
(B2) Free end	$\begin{bmatrix} 0 & -\alpha_1^{ST} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & \alpha_N^{ST} \end{bmatrix}$
(B3) End spring	$\begin{bmatrix} k_0 & -\alpha_1^{ST} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ k_N & \alpha_N^{ST} \end{bmatrix}$
(B3) End spring-mass	$\begin{bmatrix} k_0 - m_0\omega^2 & -\alpha_1^{ST} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ k_N - m_N\omega^2 & \alpha_N^{ST} \end{bmatrix}$

Apply Eqs. (13.75) and (13.76) recursively:

$$\begin{aligned} \{\eta_N(l_N)\} &= e^{[F_N]l_N}\eta_N(0) = e^{[F_N]l_N}\,[T_{N-1}]\,\{\eta_{N-1}(l_i)\} \\ &= e^{[F_N]l_N}\,[T_{N-1}]\,e^{[F_{N-1}]l_{N-1}}\,\{\eta_{N-1}(0)\} = \cdots \end{aligned}$$

which eventually leads to

$$\{\eta_N(l_N)\} = [\Phi]\,\{\eta_1(0)\} \tag{13.77}$$

where $[\Phi]$ is a 2-by-2 transition matrix given by

$$[\Phi] = e^{[F_N]l_N}\,[T_{N-1}]\,e^{[F_{N-1}]l_{N-1}}\,[T_{N-2}]\,e^{[F_{N-2}]l_{N-2}}\,\cdots\,[T_1]\,e^{[F_1]l_1} \tag{13.78}$$

Substituting Eq. (13.77) into Eq. (13.74) gives

$$\{[M_b] + [N_b]\,[\Phi]\}\,\{\eta_1(0)\} = 0 \tag{13.79}$$

The characteristic equation of the stepped system is

$$\Delta(\omega) \equiv \det\{[M_b] + [N_b]\,[\Phi]\} = 0 \tag{13.80}$$

where $\Delta(\omega)$ is a transcendental function with an infinite number of roots. The eigensolutions of the stepped system are computed in three steps. First, solve Eq. (13.80) for the natural frequencies $\omega_1, \omega_2, \ldots$ Second, for an eigenvalue ω_k, solve Eq. (13.79) for a nonzero vector $\{\eta_1(0)\}$. Third, with $\{\eta_1(0)\}$, compute the mode shape associated with ω_k by

$$v_i(x) = \left[\cos\beta_i x \quad \frac{1}{\beta_i}\sin\beta_i x\right]\{\eta_i(0)\}, \quad 0 \le x \le l_i \tag{13.81}$$

with

$$\{\eta_i(0)\} = [T_{i-1}]\,e^{[F_{i-1}]l_{i-1}}\,[T_{i-2}]\,e^{[F_{i-2}]l_{i-2}}\,\cdots\,[T_1]\,e^{[F_1]l_1}\,\{\eta_1(0)\}, \quad i > 1 \tag{13.82}$$

EXAMPLE 4.1

Consider a two-span elastic bar in Fig. 13.4.4, which has an end mass (M) and a spring constraint (k). Let the longitudinal displacements of the bar elements be

$$u_i(x,t) = v_i(x)e^{j\omega t},\ j = \sqrt{-1}, \quad i = 1,2$$

The compatibility conditions at node 1 are

$$v_1(l_1) = v_2(0)$$

$$EA_1\frac{d}{dx}v_1(l_1) + kv_1(l_1) = EA\frac{d}{dx}v_2(0)$$

and the boundary conditions at nodes 0 and 2 are

$$v_1(0) = 0$$

$$-\omega^2 M v_2(l_2) + EA_2\frac{d}{dx}v_2(l_2) = 0$$

Write the displacements of the bar elements as

$$v_i(x) = a_i\cos\beta_i x + b_i\sin\beta_i x, \quad \beta_i = \omega\sqrt{\frac{\rho_v A_i}{EA_i}}$$

and substitute them into the compatibility and boundary conditions, to obtain the following set of algebraic equations

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \cos\beta_1 l_1 & \sin\beta_1 l_1 & -1 & 0 \\ d_{31} & d_{32} & EA_2\beta_2\sin\beta_2 l_2 & -EA_2\beta_2\cos\beta_2 l_2 \\ 0 & 0 & d_{43} & d_{44} \end{bmatrix} \begin{pmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{pmatrix} = 0$$

with

$$d_{31} = -EA_1\beta_1\sin\beta_1 l_1 + k\cos\beta_1 l_1, \qquad d_{32} = EA_1\beta_1\cos\beta_1 l_1 + k\sin\beta_1 l_1$$

$$d_{43} = \omega^2 M\cos\beta_2 l_2 + EA_2\beta_2\sin\beta_2 l_2, \qquad d_{44} = \omega^2 M\sin\beta_2 l_2 - EA_2\beta_2\cos\beta_2 l_2$$

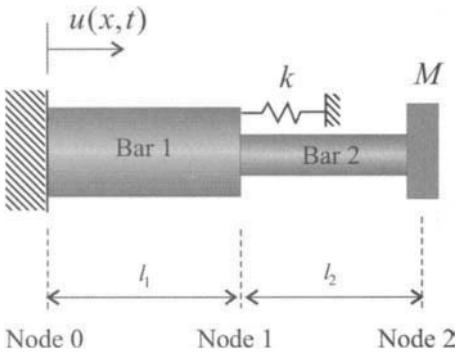


FIGURE 13.4.4

Solution of the above homogeneous equations yields the natural frequencies and mode shapes of the stepped bar.

If the eigenvalue problem is solved by the DTFM, the transition matrix and boundary condition matrices are formed as follows

$$[\Phi] = \begin{bmatrix} \cos\beta_2 l_2 & \frac{1}{\beta_2}\sin\beta_2 l_2 \\ -\beta_2 \sin\beta_2 l_2 & \cos\beta_2 l_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{k}{EA_2} & \frac{EA_1}{EA_2} \end{bmatrix} \begin{bmatrix} \cos\beta_1 l_1 & \frac{1}{\beta_1}\sin\beta_1 l_1 \\ -\beta_1 \sin\beta_1 l_1 & \cos\beta_1 l_1 \end{bmatrix}$$

$$[M_b] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad [N_b] = \begin{bmatrix} 0 & 0 \\ -M\omega^2 & EA_2 \end{bmatrix}.$$

With these matrices, the solution of Eq. (13.79) leads to the natural frequencies and mode shapes. (See Section 13.4.3.)

13.4.3 Solution by the Toolbox

The Toolbox of this chapter offers several functions for computing and displaying the eigensolutions of a stepped distributed system and for animating its modes of vibration. To compute the natural frequencies and mode shapes of a stepped system, call one of functions `setbar2`, `setshaft2`, and `setstring2`; see Windows 4.1.

Window 4.1. Functions `setbar2`, `setshaft2`, and `setstring2`

MATLAB Functions: **`setbar2, setshaft2, setstring2`**

Purpose:
To compute the eigensolutions of a stepped system.

Synopsis:

```
setbar2(Bar_Spec, BC_Spec, Constr_Spec, n_mode)
setshaft2(Shaft_Spec, BC_Spec, Constr_Spec, n_mode)
setstring2(T, Element_Spec, BC_Spec, Constr_Spec, n_mode)
```

Description:
`setbar2(Bar_Spec, BC_Spec, Constr_Spec, n_mode)` computes the natural frequencies and associate mode shapes of a stepped elastic bar in longitudinal vibration, where `Bar_Spec` is a matrix specifying the parameters of bar elements, `BC_Spec` is a vector specifying the boundary conditions of the structure, `Constr_Spec` is a matrix specifying the constraints of the structure at interior nodes (nodes 1 to N-1 in Fig. 13.4.1), and `n_mode` is the total number of modes selected for computation. By default, `n_mode = 8`. If the bar structure has no constrains, set `Constr_Spec = 0` or replace it by `[ ]` as a placeholder. For instance, `setbar2(Bar_Spec, BC_Spec)` specifies a stepped shaft with no constraints at its interior nodes.

`setshaft2(Shaft_Spec, BC_Spec, Constr_Spec, n_mode)` computes the natural frequencies and associate mode shapes of a stepped circular shaft in torsional vibration, where `Shaft_Spec` is a matrix specifying the parameters of shaft elements and other input arguments are similar to those of function `setbar2`.

Window 4.1. Functions setbar2, setshaft2, and setstring2 (*Continued*)

`setstring2(T, Element_Spec, BC_Spec, Constr_Spec, n_mode)` computes the natural frequencies and associate mode shapes of a stepped taut string in lateral vibration, where `T` is the tension of the string, `Element_Spec` is a matrix specifying the parameters of string segments, and other input arguments are similar to those of function `setbar2`.

The formation of the input arguments `Bar_Spec`, `Shaft_Spec`, `Element_Spec`, `BC_Spec`, and `Constr_Spec` are described following this Window.

Functions `setbar2`, `setshaft2`, and `setstring2` assign system parameters, boundary conditions, and constraints by the following arguments:

(i) The parameters of an N-element bar are assigned by an $N \times 3$ matrix

$$\texttt{Bar\_Spec} = \begin{bmatrix} \rho_v A_1 & EA_1 & L_1 \\ \rho_v A_2 & EA_2 & L_2 \\ \vdots & \vdots & \\ \rho_v A_N & EA_N & L_N \end{bmatrix} \tag{13.83}$$

where the ith row contains the linear density, longitudinal rigidity, and length of the ith bar element. Likewise, the parameters of an N-segment shaft are assigned by an $N \times 3$ matrix

$$\texttt{Shaft\_Spec} = \begin{bmatrix} \rho_v J_1 & GJ_1 & L_1 \\ \rho_v J_2 & GJ_2 & L_2 \\ \vdots & \vdots & \vdots \\ \rho_v J_N & GJ_N & L_N \end{bmatrix} \tag{13.84}$$

where the ith row contains the linear density, torsional rigidity, and length of the ith shaft element. The parameters of an N-segment string are assigned by an $N \times 2$ matrix

$$\texttt{Element\_Spec} = \begin{bmatrix} \mu_1 & L_1 \\ \mu_2 & L_2 \\ \vdots & \vdots \\ \mu_N & L_N \end{bmatrix} \tag{13.85}$$

where the ith row contains the linear density and length of the ith string segment.

(ii) The boundary conditions of a stepped system are assigned by a vector

```
BC_Spec = [BC_Left  BC_Right]
```
(13.86)

where `BC_Left` and `BC_Right` are two-row subvectors describing the boundary conditions at the left end (Node 0) and the right end (Node N) of the stepped system, according to Table 13.4.1. For instance, the boundary conditions of a stepped system with its left end fixed and right end constrained by an end spring of coefficient k_N and a lumped mass m_N are described by

```
BC_Spec = [1 4 k_N m_N].
```

(iii) The constraints at the interior nodes of a stepped system are described by a matrix

$$\texttt{Constr\_Spec} = \begin{bmatrix} \text{Constr\_1} \\ \text{Constr\_2} \\ \vdots \\ \text{Constr\_Nc} \end{bmatrix} \tag{13.87}$$

where Nc is the number of constraints. The jth row of Constr_Spec specifies the jth constraint by

$$\texttt{Constr\_j = [node\_no kc mc]} \tag{13.88}$$

where node_no is node number (1, 2,..., N-1), kc is the coefficient of a spring constraint imposed at the node, and mc is a lumped mass mounted at the node. For example, for the stepped shaft shown in Fig. 13.4.2.

$$\texttt{Constr\_Spec} = \begin{bmatrix} 1 & 0 & I_D \\ 3 & k_\tau & 0 \end{bmatrix}$$

where I_D is the moment of inertia of the rigid disk mounted at node 1 and k_τ is the coefficient of the torsional spring at node 3.

The information on the system parameters, boundary conditions, and constraints that are assigned by function setbar2 or setshaft2 or setstring2 can be retrieved by typing systinfo2 in the MATLAB command window.

The mode shapes of a stepped system that are obtained by the functions in Window 4.1 can be plotted by function plotmsh2; see Window 4.2. After a stepped system is set up by one of the functions in Window 4.1, the eigenvalues and normalized eigenfunctions of the system can also be computed by function eigenstep2; see Window 4.3. In addition, function animode2 can be used to animate the vibration of a stepped system in a particular mode; see Window 4.4.

Window 4.2. Function plotmsh2

MATLAB Function: **plotmsh2**

Purpose:
To plot a mode shape of a stepped system that is computed by one of the functions setbar2, setshaft2, and setstring2.

Synopsis:

```
plotmsh2(n)
plotmsh2(n, opt, npts)
[y, x] = plotmsh2(n, opt, npts)
```

Window 4.2. Function plotmsh2 (*Continued*)

Description:
`plotmsh(n)` plots the nth mode shape of a stepped system over its length.

`plotmsh(n, opt, npts)` plots the spatial distribution of nth mode with the options:

`opt = 0`	plot modal displacement
`opt = 1`	plot modal internal force
`opt = 2`	plot both modal displacement and internal force

Here `npts` is the number of equally-spaced points along the length of each element of the stepped system that are selected for computation. By default, `opt` = 0 and `npts` = 101.

`[y, x] = plotmsh2(n, opt, npts)` returns the modal displacement and internal force in a two-row vector `y` and spatial points of computation in a vector `x`. With `x` and `y`, the modal displacement and internal force can be plotted by `plot(x, y(1,:))` and `plot(x, y(2,:))`.

Window 4.3. Function eigenstep2

MATLAB Function: **eigenstep2**

Purpose:
To determine the eigenvalues and normalized eigenfunctions of a stepped distributed dynamic system.

Synopsis:

```
[y, msh] = eigenstep2(n)
```

Description:
`[y, msh] = eigenstep2(n)` returns the first `n` eigenvalues of the system in a vector `y` and the coefficients of normalized eigenfunctions in a three-dimensional matrix `msh`. The kth page of the matrix `msh`, `msh(:,:, k)` contains the parameters of the kth eigenfunction as follows:

For a flexible mode, the displacement of the jth element is $v(x) = a_1 \cos \beta x + a_2 \sin \beta x$ with

$$\texttt{msh(1, j, k)} = \beta, \quad \texttt{msh(2, j, k)} = a_1, \quad \texttt{msh(3, j, k)} = a_2$$

For a rigid-body mode, the displacement of the jth element is $v(x) = a$ with

$$\texttt{msh(1, j, k)} = 0, \quad \texttt{msh(2, j, k)} = \texttt{msh(3, j, k)} = a$$

Window 4.4. Function animode2

MATLAB Function: **animode2**

Purpose:
To animate a mode of vibration of a stepped system in its spatial region.

Synopsis:

```
animode2(k)
F = animode2(k, tf, n_frames)
```

Description:
`animode2(k)` animates the time response of the system in the kth mode of vibration.

`F = animode2(k, tf, n_frames)` returns a set of frames on the time response of the kth mode in a matrix `F`, which can be played back by the MATLAB function `movie(F)`. Here `tf` is total animation time, and `n_frames` is the number of frames in the animation. By default, `tf` = 8*T, with T being the period of the mode, and `n_ frames` = 97.

EXAMPLE 4.2

Let the parameters of the stepped bar in Example 4.1 be

$$\rho_v A_1 = 0.5, \quad EA_1 = 100, \quad L_1 = 1$$

$$\rho_v A_2 = 0.7, \quad EA_1 = 15, \quad L_1 = 2$$

$$k = 250, \quad M = 20$$

```
>> Bar_Spec = [0.5 100 1; 0.7 150 2];
>> BC_Spec = [1 4 0 20];
>> Constr_Spec = [1 250 0];
>> setbar2(Bar_Spec, BC_Spec, Constr_Spec)
```

yields the natural frequencies of the stepped system as follows

```
First 8 Eigenvalues
-------------------

    wn          fn
  1.7327      0.2758
 18.8123      2.9941
 34.1503      5.4352
 45.3883      7.2238
 61.7371      9.8258
 77.8369     12.3881
 90.7058     14.4363
106.6238     16.9697
```

```
where
wn = natural frequencies in rad/s
fn = wn/(2*pi), natural frequencies in Hertz
```

and the mathematical expressions of the normalized eigenfunctions (not shown here). The first two mode shapes are plotted in Figs. 13.4.5 and 13.4.6 by

```
>> plotmsh2(1)
```

and

```
>> plotmsh2(2)
```

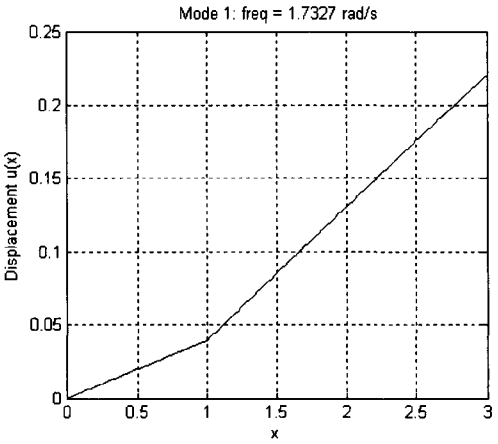


FIGURE 13.4.5 The first mode shape.

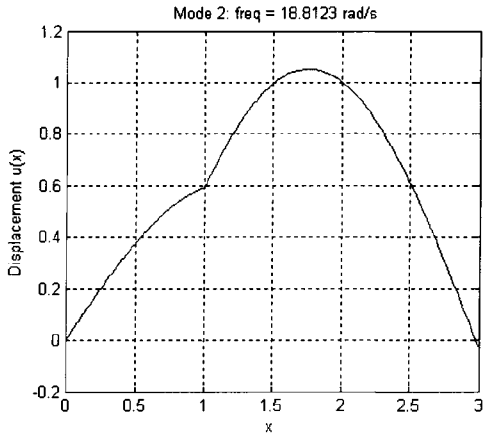


FIGURE 13.4.6 The second mode shape.

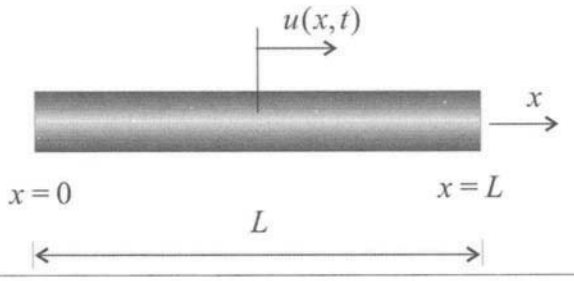
13.5 Quick Solution Guide

Type I—Single-Body Systems

Elastic bar in longitudinal deformation

$$\rho_v A \frac{\partial^2}{\partial t^2} u(x,t) - EA \frac{\partial^2}{\partial x^2} u(x,t) = q(x,t)$$

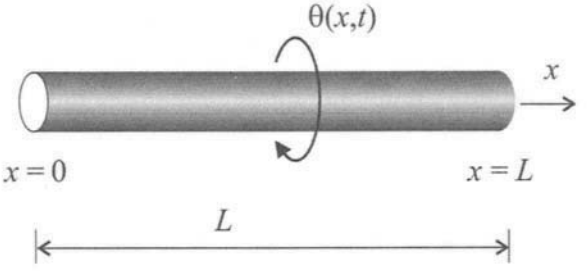
$u(x,t)$: longitudinal displacement
$q(x,t)$: external load
ρ_v : material density (mass per unit volume)
E : Young's modulus
A : cross-section area



Circular shaft in torsional deformation

$$\rho_v J \frac{\partial^2}{\partial t^2} \theta(x,t) - GJ \frac{\partial^2}{\partial x^2} \theta(x,t) = \tau(x,t)$$

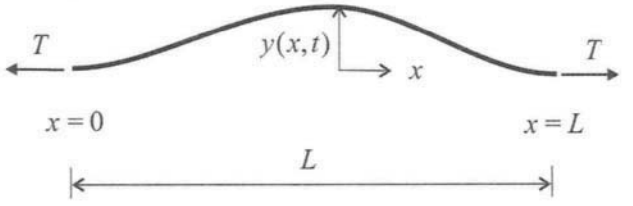
$\theta(x,t)$: rotation (twist)
$\tau(x,t)$: external torque
ρ_v : material density (mass per unit volume)
G : shear modulus
J : polar moment of inertia



Taut string with lateral deflection

$$\mu \frac{\partial^2}{\partial t^2} y(x,t) - T \frac{\partial^2}{\partial x^2} y(x,t) = p(x,t)$$

$y(x,t)$: lateral displacement
$p(x,t)$: lateral external load
μ : linear density (mass per unit length)
T : string tension

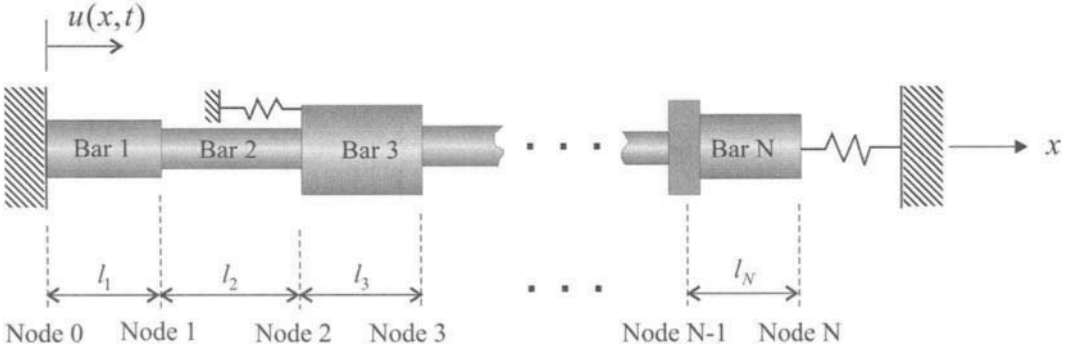


Problems of Single-Body Systems To Solve

(a) Eigenvalue problems (natural frequencies and mode shapes);
(b) Free vibration and forced vibration; and
(c) Frequency response.

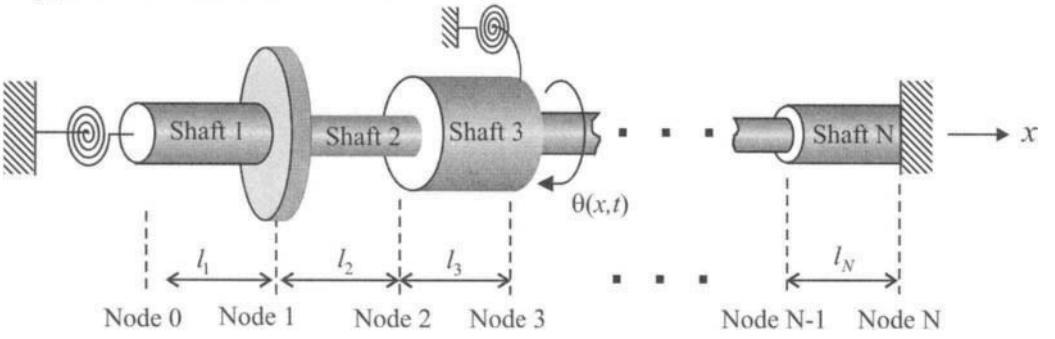
Type II—Stepped Systems

Stepped elastic bar in longitudinal vibration

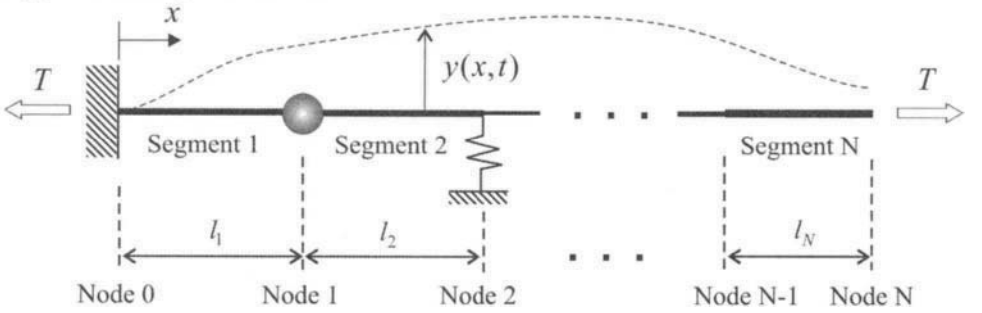


(*continued*)

Stepped circular shaft in torsional vibration



Stepped taut string in lateral vibration



Problem of Stepped Systems To Solve

Eigenvalue problems (natural frequencies and mode shapes).

MATLAB Functions

This chapter has a set (toolbox) of MATLAB functions for dynamic analysis of elastic bars, torsional shafts, and taut strings; see the table below. Refer to the corresponding window or section for the utility of each function.

	System Setup	
`setbar`	Set up a bar and compute its eigensolutions	Window 2.1
`setshaft`	Set up a shaft and compute its eigensolutions	Window 2.1
`setstring`	Set up a string and compute its eigensolutions	Window 2.1
`systinfo`	Display the information on system parameters, boundary conditions, and eigensolutions	Section 13.2.3
	Eigensolutions	
`eigen2os`	Determine eigenvalues and normalized eigenfunctions	Window 2.2
`nodalpts`	Determine the nodal points of eigenfunctions	Window 2.3
`plotmsh`	Plot spatial distribution of eigenfunctions	Window 2.4
`animode`	Animate modes of vibration	Window 2.5

Dynamic Response		
`free2os`	Plot the free response of a second-order system	Window 3.1
`free2os1`		
`forced2os`	Plot the forced response of a second-order system	Window 3.2
`forced2os1`		
`frf2os`	Plot the frequency response of a second-order system	Window 3.5
`movie2os`	Animate time response due to external and initial disturbances	Window 3.4
Free Vibration of Stepped Systems		
`setbar2`	Set up a stepped bar and compute its eigensolutions	Window 4.1
`setshaft2`	Set up a stepped shaft and compute its eigensolutions	Window 4.1
`setstring2`	Set up a stepped string and compute its eigensolutions	Window 4.1
`systinfo2`	Display the information on system parameters, boundary conditions, and eigensolutions	Section 13.4.3
`plotmsh2`	Plot spatial distribution of eigenfunctions of a stepped system	Window 4.2
`eigenstep2`	Obtain eigenfunctions of a stepped system	Window 4.3
`animode2`	Animate modes of vibration for a stepped system	Window 4.4
Utilities		
`TBdemo`	Show how the Toolbox works and what it can do	Section 13.1
`TBinfo`	Show the information of the functions in the Toolbox	Section 13.1
`RunEx`	Run all numerical examples in this chapter	Section 13.1

Solution Procedure

For dynamic analysis of a single-body system, type the following commands in the MATLAB command window.

Step 1. Set up a system and compute eigensolutions

For an elastic bar, type

```
>> setbar(rou, EA, L, BC_Spec)
```

For a circular shaft, type

```
>> setshaft(rou, GJ, L, BC_Spec)
```

For a taut string, type

```
>> setstring(rou, T, L, BC_Spec)
```

Step 2. Compute dynamic response

To compute free vibration at position `xr`, type

```
>> free2os(IC, xr)
```

To compute forced vibration at position `xr`, type

```
>> forced2os(Load_Spec, xr)
```

To compute frequency response at position `xr`, type

```
>> frf2os(xr, xf, f0)
```

where `xf` and `f0` are the location and amplitude of a sinusoidal load, respectively.

Step 3. Display dynamic response

To plot spatial distribution of the kth mode, type

```
>> plotmsh(k)
```

To animate the kth mode of vibration, type

```
>> animode(k)
```

To animate time response, type

```
>> movie2os(Load_Spec, IC)
```

The `BC_Spec` is a vector specifying the boundary conditions; `IC` is a two-row matrix specifying initial conditions; and `Load_Spec` is a vector specifying an external load. The formation of the vectors and matrix is given below.

(1) Specification of boundary conditions by vector `BC_Spec`

Boundary Conditions	**BC_Spec = [BC_Left BC_Right]**	
	Left Boundary (x = 0) `BC_Left`	*Right Boundary (x = L)* `BC_Right`
(B1) Fixed end At $x = 0$: $w(0,t) = 0$ At $x = L$: $w(L,t) = 0$	[1]	[1]
(B2) Free end At $x = 0$: $\frac{\partial}{\partial x} w(0,t) = 0$ At $x = L$: $\frac{\partial}{\partial x} w(L,t) = 0$	[2]	[2]
(B3) End spring At $x = 0$: $k_s w(0,t) - \alpha_{ST} \frac{\partial}{\partial x} w(0,t) = 0$ At $x = L$: $k_s w(L,t) + \alpha_{ST} \frac{\partial}{\partial x} w(L,t) = 0$ (k_s - spring coefficient)	[3 k_s]	[3 k_s]

(2) Specification of initial conditions by matrix `IC`

Given initial displacement $w(x,0) = a_0(x)$ and initial velocity $\frac{\partial}{\partial t} w(x,0) = b_0(x)$, the matrix is formed as

$$\mathtt{IC} = \begin{bmatrix} a_0(x_1) & a_0(x_2) & \cdots & a_0(x_{np}) \\ b_0(x_1) & b_0(x_2) & \cdots & b_0(x_{np}) \end{bmatrix}$$

where x_k, $k = 1, 2, \ldots, np$ are np equally-spaced points in the spatial domain $[0, L]$.

(3) Specification of external loads by vector Load_Spec

External Load	*F(x, t)*	Load_Spec
(L0) Pointwise impulse load	$I_0\delta(t)\delta(x-x_f)$	[0 x_f I_0]
(L1) Pointwise step load	$F_0\delta(x-x_f)$	[1 x_f F_0]
(L2) Pointwise sinusoidal load	$A\sin(\omega t+\phi_0)\delta(x-x_f)$	[2 x_f A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
(L10) Uniformly distributed impulse load	$I_0\delta(t)$, for $x_1 \le x \le x_1$	[10 x_1 x_2 I_0]
(L11) Uniformly distributed step load	F_0, for $x_1 \le x \le x_1$	[11 x_1 x_2 F_0]
(L12) Uniformly distributed sinusoidal load	$A\sin(\omega t+\phi_0)$, for $x_1 \le x \le x_1$	[12 x_1 x_2 A ω ϕ_0] ω in rad/s, ϕ_0 in degrees

For free vibration analysis of a stepped system, type the following commands in the MATLAB command window:

Step 1. Compute natural frequencies and mode shapes

For a stepped elastic bar, type

```
>> setbar2(Bar_Spec, BC_Spec, Constr_Spec)
```

For a stepped circular shaft, type

```
>> setshaft2(Shaft_Spec, BC_Spec, Constr_Spec)
```

For a stepped taut string, type

```
>> setstring2(T, Element_Spec, BC_Spec, Constr_Spec)
```

Step 2. Display eigensolutions

To plot spatial distribution of the *k*th mode, type

```
>> plotmsh(k)
```

To animate the *k*th mode of vibration, type

```
>> animode2(k)
```

13.6 References

1. Den Hartog J P 1985 *Mechanical Vibrations*, Dover Publications: New York.
2. Inman D J 2000 *Engineering Vibrations*, 2nd edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
3. Meirovitch L 1967 *Analytical Methods in Vibrations*, Macmillan: New York.
4. Rao S S 2003 *Mechanical Vibrations*, 4th edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
5. Timoshenko S 1990 *Vibration Problems in Engineering*, 5th edition, John Wiley & Sons.

This Page Intentionally Left Blank

14 Dynamic Analysis of Constrained, Combined, and Stepped Beams

Inside
- Getting Started
- Constrained Beams
- Combined Beams
- Stepped Beams
- Quick Solution Guide
- References

14.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for modeling and dynamic analysis of constrained beams, combined beams, and stepped beams.

System Requirements for the MATLAB Toolbox

- PC with Win 98SE/NT/2000 and XP or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed on the computer

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

At this stage, the Toolbox is properly installed, and it is ready for use.

Quick Tutorial

To show how to use the Toolbox, consider a clamped-pinned beam combined with an oscillator, as shown in Fig. 14.1.1. The beam parameters are as follows: linear density $\rho = 2.5$, bending stiffness $EI = 100$, and length $L = 4$. Let the oscillator parameters be: location $a = 3$, spring coefficient $k = 80$, and mass $m = 2$. By the following commands typed after the prompt >> in the MATLAB command window:

```
>> rou = 2.5; EI = 100; L = 4;
>> a = 3; k = 80; m = 2; Osc_Spec = [a k m 0];
>> setbeam2(rou, EI, L, 0, [2 1], 0, Osc_Spec)
```

the first nine natural frequencies of the combined beam are computed as follows

```
Eigenvalues of the Combined Beam Structure
-----------------------------------------
The system is undamped with natural frequencies:
   mode 1: omega = 4.6775 rad/sec
   mode 2: omega = 8.0425 rad/sec
   mode 3: omega = 20.1875 rad/sec
   mode 4: omega = 41.27 rad/sec
   mode 5: omega = 70.4717 rad/sec
   mode 6: omega = 107.5814 rad/sec
   mode 7: omega = 152.4454 rad/sec
   mode 8: omega = 205.0744 rad/sec
   mode 9: omega = 265.5336 rad/sec
```

and the first four mode shapes of the beam are plotted in Fig. 14.1.2.

Furthermore, assume that a harmonic force is applied to the beam at location $x_f = 3$. The steady-state displacement (frequency response) of the beam at $x_r = 2$ is obtained by the following commands:

```
>> xr = 2.0; xf = 3.0;
>> frfbeam2(xr, xf, 1, 0, 80)
```

See Fig. 14.1.3 for the frequency response.

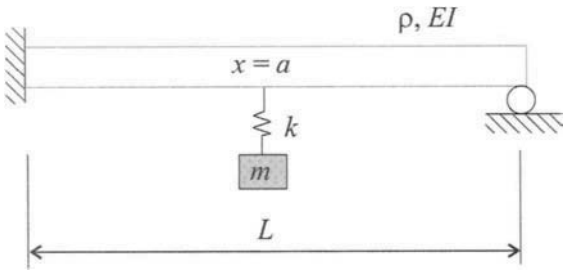


FIGURE 14.1.1 A combined beam.

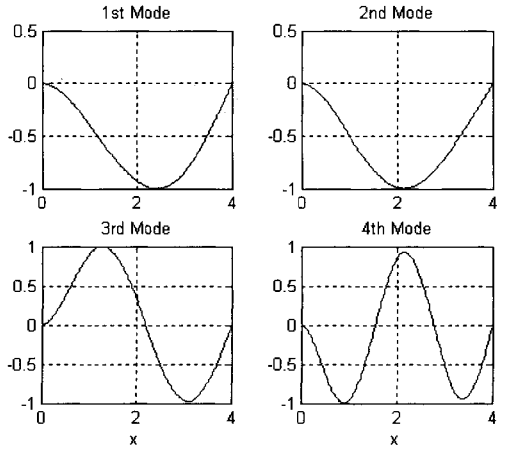


FIGURE 14.1.2 The mode shapes of the combined beam.

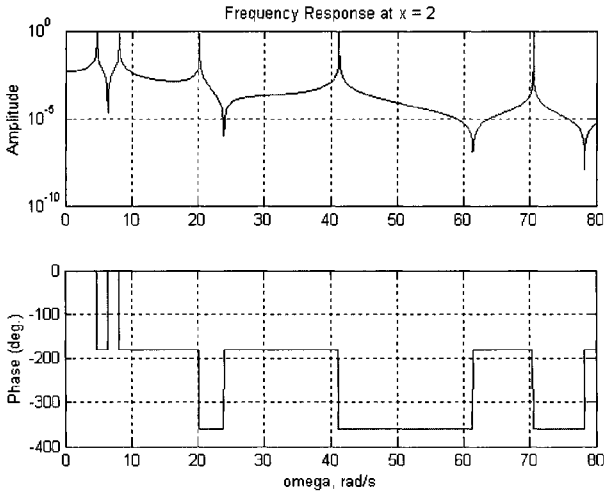


FIGURE 14.1.3 The frequency response of the combined beam.

In the above simulation, two functions from the Toolbox are called:

(a) Function **setmbeam2** that inputs system parameters, sets up boundary conditions, and computes the natural frequencies and mode shapes of the combined beam; and
(b) Function **frfbeam2** that plots the frequency response of the combined beam.

The above functions, along with others from the Toolbox, will be discussed in detail in the subsequent sections.

For a quick solution, the user may refer directly to the Quick Solution Guide (Section 14.5) or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window. To better understand how the toolbox works, the user is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 14.6.

14.2 Constrained Beams

A constrained beam in consideration is a uniform Euler-Bernoulli beam with p sets of attached springs, lumped masses, and dampers (SMD); see Fig. 14.2.1, where x_i and k_i, m_i, c_i are the location, and the spring, mass, and damping coefficients of the ith SMD set, respectively.

This chapter offers a toolbox of MATLAB functions for modeling and dynamic analysis of constrained beams. Solution by the Toolbox takes the following two steps:

Step 1. Set up system parameters and boundary conditions

To set up a constrained beam, call function `setbeam`. Function `setbeam` also yields the eigensolutions of the beam.

Step 2. Compute and display system response

To plot mode shapes, call function `plotmsh`.

To animate modes of vibration, call function `animode`.

To plot eigenvalue loci, call function `eiglocus`

To compute frequency response, call function `frfbeam`.

These functions, along with others from the Toolbox, will be presented in sequel.

14.2.1 System Description

Governing Equations The transverse displacement $w(x,t)$ of the constrained beam shown in Fig. 14.2.1 is governed by the partial differential equation (References 3 and 5)

$$\rho \frac{\partial^2}{\partial t^2} w(x,t) + d_v \frac{\partial}{\partial t} w(x,t) + EI \frac{\partial^4}{\partial x^4} w(x,t) = F(x,t) - f_c(x,t), \quad 0 \le x \le L \tag{14.1}$$

with initial conditions and appropriate boundary conditions, where ρ, EI, and L are the linear density (mass per unit length), bending stiffness, and length of the beam, respectively; d_v is a

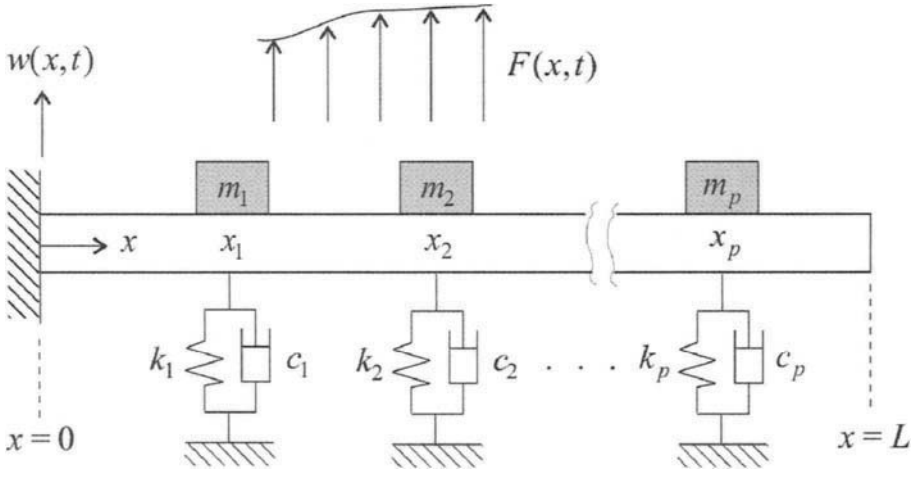


FIGURE 14.2.1 Schematic of a constrained beam.

viscous damping coefficient; and $F(x,t)$ is the external force. The $f_c(x,t)$ represents constraint forces exerted to the beam by the attached SMDs, and is given by

$$f_c(x,t) = \sum_{i=1}^{p} \delta(x - x_i)\left(m_i \frac{\partial^2}{\partial t^2} + c_i \frac{\partial}{\partial t} + k_i\right) w(x,t) \tag{14.2}$$

where $\delta(x-x_i)$ is the delta function. Besides viscous damping, modal damping is also modeled. In many applications, the external force in Eq. (14.1) can be written as

$$F(x,t) = b(x) \cdot f(t) \tag{14.3}$$

where function $b(x)$ describes the spatial distribution of the force. Table 14.2.1 lists seven types of boundary conditions, of which the first four (B1 to B4) are classical boundary conditions. The slope (rotation), bending moment, and shear force of the beam are

$$\theta(x,t) = \frac{\partial}{\partial x} w(x,t), \quad M(x,t) = EI \frac{\partial^2}{\partial x^2} w(x,t), \quad Q(x,t) = EI \frac{\partial^3}{\partial x^3} w(x,t) \tag{14.4}$$

Modal Expansion Equation (14.1) is solved by eigenfunction expansion or modal expansion. To this end, the transverse displacement $w(x,t)$ of the beam is approximated by a modal series of n terms

$$w(x,t) \approx \sum_{k=1}^{n} \phi_k(x) q_k(t) = [\Phi(x)]^T \{q(t)\} \tag{14.5}$$

where

$$[\Phi(x)] = (\phi_1(x)\ \phi_2(x) \cdots \phi_n(x))^T$$
$$\{q(t)\} = (q_1(t)\ q_2(t) \cdots q_n(t))^T .$$

The $q_k(t)$ are unknown generalized coordinates, and $\phi_k(x)$ are the eigenfunctions of the "unconstrained" beam, which is the beam in Fig. 14.2.1 without damping and attached SMDs. In other words, $\phi_k(x)$ are the solutions of

$$\rho\alpha^2 \phi(x) = EI \frac{d^4}{dx^4} \phi(x), \quad 0 \le x \le L \tag{14.6}$$

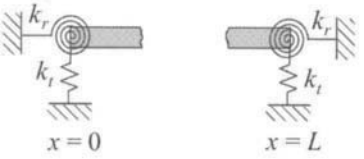
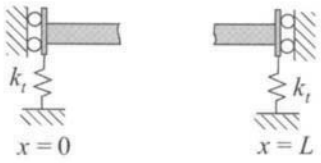
subject to the same boundary conditions as those of the constrained beam, where α is a natural frequency of the unconstrained beam. The eigenfunctions satisfy the orthonormal conditions

$$\begin{aligned} \rho \int_0^L \phi_j(x)\phi_k(x)dx &= \delta_{jk} \\ EI \int_0^L \phi_j(x) \frac{d^4}{dx^4} \phi_k(x)dx &= \delta_{jk}\alpha_k^2 \end{aligned} \tag{14.7}$$

where the delta function δ_{jk} is one for $j = k$, and zero otherwise. For eigensolutions of Eq. (14.6) refer to Chapter 12.

TABLE 14.2.1 Boundary Conditions of Beams in Transverse Vibration

(k_r - torsional spring coefficient, k_t - translational spring coefficient)

Boundary Type	*Boundary Conditions*
(B1) Pinned or hinged end	Left end: $w(0,t) = 0, \quad -EI\dfrac{\partial^2 w(0,t)}{\partial x^2} = 0$ Right end: $w(L,t) = 0, \quad EI\dfrac{\partial^2 w(L,t)}{\partial x^2} = 0$
(B2) Clamped or fixed end	Left end: $w(0,t) = 0, \quad \dfrac{\partial w(0,t)}{\partial x} = 0$ Right end: $w(L,t) = 0, \quad \dfrac{\partial w(L,t)}{\partial x} = 0$
(B3) Free end	Left end: $-EI\dfrac{\partial^2 w(0,t)}{\partial x^2} = 0, \quad EI\dfrac{\partial^3 w(0,t)}{\partial x^3} = 0$ Right end: $EI\dfrac{\partial^2 w(L,t)}{\partial x^2} = 0, \quad -EI\dfrac{\partial^3 w(L,t)}{\partial x^3} = 0$
(B4) Sliding or guided end	Left end: $\dfrac{\partial w(0,t)}{\partial x} = 0, \quad EI\dfrac{\partial^3 w(0,t)}{\partial x^3} = 0$ Right end: $\dfrac{\partial w(L,t)}{\partial x} = 0, \quad -EI\dfrac{\partial^3 w(L,t)}{\partial x^3} = 0$
(B5) Elastic-support 	Left end: $k_r\dfrac{\partial w(0,t)}{\partial x} - EI\dfrac{\partial^2 w(0,t)}{\partial x^2} = 0, \quad k_t w(0,t) + EI\dfrac{\partial^3 w(0,t)}{\partial x^3} = 0$ Right end: $k_r\dfrac{\partial w(L,t)}{\partial x} + EI\dfrac{\partial^2 w(L,t)}{\partial x^2} = 0, \quad k_t w(L,t) - EI\dfrac{\partial^3 w(L,t)}{\partial x^3} = 0$
(B6) Hinged-elastic end	Left end: $w(0,t) = 0, \quad k_r\dfrac{\partial w(0,t)}{\partial x} - EI\dfrac{\partial^2 w(0,t)}{\partial x^2} = 0$ Right end: $w(L,t) = 0, \quad k_r\dfrac{\partial w(L,t)}{\partial x} + EI\dfrac{\partial^2 w(L,t)}{\partial x^2} = 0$
(B7) Sliding-elastic end 	Left end: $\dfrac{\partial w(0,t)}{\partial x} = 0, \quad k_t w(0,t) + EI\dfrac{\partial^3 w(0,t)}{\partial x^3} = 0$ Right end: $\dfrac{\partial w(L,t)}{\partial x} = 0, \quad k_t w(L,t) - EI\dfrac{\partial^3 w(L,t)}{\partial x^3} = 0$

Substitution of the modal series (14.5) into Eq. (14.1) and use of the orthonormal conditions (14.7) leads to the matrix differential equation for the constrained beam

$$[M]\{\ddot{q}(t)\} + [D]\{\dot{q}(t)\} + [K]\{q(t)\} = \{B_f\} f(t) \tag{14.8}$$

with

$$[M] = [I] + [M_c], \quad [D] = [D_b] + [D_c], \quad [K] = [\Omega_b] + [K_c] \tag{14.9}$$

where $[I]$ is an identity matrix; $\{B_f\}$ is a force distribution vector

$$\{B_f\} = \int_0^L [\Phi(x)] b(x) dx;$$

and

$$[D_b] = \begin{cases} \dfrac{d_v}{\rho}[I] & \text{for the beam with viscous damping} \\ \underset{i}{\text{diag}}\{2\xi_i \alpha_i\} & \text{for the beam with modal damping} \end{cases}$$

$$[\Omega_b] = \underset{k}{\text{diag}}\left\{\alpha_k^2\right\}; \quad [M_c] = \sum_{j=1}^{p} m_j \left[\Phi(x_j)\right]\left[\Phi(x_j)\right]^T$$

$$[D_c] = \sum_{j=1}^{p} c_j \left[\Phi(x_j)\right]\left[\Phi(x_j)\right]^T; \quad [K_c] = \sum_{j=1}^{p} k_j \left[\Phi(x_j)\right]\left[\Phi(x_j)\right]^T$$

In the previous expressions, ξ_i and α_i are the damping ratio and natural frequency of the ith mode of the unconstrained beam. Physically, $[D_b]$ and $[\Omega_b]$ describe the damping and stiffness of the beam, while $[M_c]$, $[D_c]$, and $[K_c]$ describe the effects of the attached SMDs. Equation (14.8) is a discretized model of the constrained beam.

Eigenvalue Problem The eigenvalue problem of a constrained beam, by Eq. (14.8), is described by

$$\left(\lambda^2 [M] + \lambda [D] + [K]\right)\{v\} = 0 \tag{14.10}$$

where λ is an eigenvalue and $\{v\}$ is the associate eigenvector. The characteristic equation is

$$\det\left(\lambda^2 [M] + \lambda [D] + [K]\right) = 0 \tag{14.11}$$

the roots of which, λ_k, are the eigenvalues of the constrained beam. With the eigensolutions, the kth mode of vibration of the constrained beam is expressed as

$$w_k(x,t) = \psi_k(x)\, e^{\lambda_k t}, \quad 0 \le x \le L \tag{14.12}$$

where the mode shape function $\psi_k(x)$ is of the form

$$\psi_k(x) = [\Phi(x)]^T \{v_k\} \tag{14.13}$$

with $[\Phi(x)]$ given in Eq. (14.5).

The format of the above eigensolutions depends on the damping matrix $[D]$. Three typical cases are examined as follows.

Case 1. Undamped System The beam is undamped and all the SMD sets have no damping components in them; namely, $[D] = 0$. Under the circumstances, Eq. (14.10) becomes

$$\omega^2 [M] \{u\} = [K] \{u\} \tag{14.14}$$

where $\lambda^2 = -\omega^2$ has been used. The corresponding characteristic equation is

$$\det\left(-\omega^2 [M] + [K]\right) = 0 \tag{14.15}$$

which has n nonnegative roots $0 \leq \omega_1^2 \leq \omega_2^2 \leq \cdots \leq \omega_n^2$. The parameters $\omega_1, \omega_2, \ldots, \omega_n$ are called the *natural frequencies* of the system. The associate eigenvectors $\{u_1\}, \{u_2\}, \ldots, \{u_n\}$ are real, and enjoy the orthogonality relations (after proper scaling)

$$\{u_j\}^T [M] \{u_k\} = \delta_{jk}, \quad \{u_j\}^T [K] \{u_k\} = \delta_{jk} \omega_k^2 \tag{14.16}$$

Case 2. Proportionally Damped System The parameter matrices of a proportionally damped system satisfy the condition (Reference 6)

$$[D] [M]^{-1} [K] = [K] [M]^{-1} [D] \tag{14.17}$$

Under this condition, all the parameter matrices can be diagonalized as

$$[U]^T [M] [U] = [I], \quad [U]^T [D] [U] = \operatorname*{diag}_{1 \leq k \leq n} \{\hat{c}_k\}, \quad [U]^T [K] [U] = \operatorname*{diag}_{1 \leq k \leq n} \left\{\omega_k^2\right\} \tag{14.18}$$

where $[U]$ is the modal matrix of the corresponding undamped system

$$[U] = \left[\{u_1\} \quad \{u_2\} \quad \cdots \quad \{u_n\}\right] \in R^{n \times n} \tag{14.19}$$

with $\{u_k\}$ being the solutions of Eq. (14.14). The eigenvectors of such a damped system are the same as $\{u_k\}$ of the undamped system; the eigenvalues related to $\{u_k\}$ are the roots of

$$\lambda^2 + \hat{c}_k \lambda + \omega_k^2 = 0$$

namely,

$$\lambda_{\pm k} = \frac{1}{2}\left(-\hat{c}_k \pm j\sqrt{4\omega_k^2 - \hat{c}_k^2}\right), \quad j = \sqrt{-1} \tag{14.20}$$

As an example, a viscously damped beam with attached springs is a proportionally damped system because its parameter matrices

$$[M] = [I], \quad [D] = \frac{d_v}{\rho} [I], \quad [K] = [\Omega_b] + [K_c] \tag{14.21}$$

TABLE 14.2.2 Eigensolutions of a Beam with Attached SMDs

Damping Status	*Eigenvalues*	*Eigenvectors*
No damping $[C] = 0$	n real and nonnegative eigenvalues $0 \le \omega_1^2 \le \omega_2^2 \le \cdots \le \omega_n^2$	n real orthogonal eigenvectors $\{u_1\}, \{u_2\}, \ldots, \{u_n\}$
Proportional damping $[C][M]^{-1}[K] = [K][M]^{-1}[C]$	$2n$ complex eigenvalues $\lambda_{\pm k} = \frac{1}{2}\left(-\hat{c}_k \pm i\sqrt{4\omega_k^2 - \hat{c}_k^2}\right)$	n real orthogonal eigenvectors $\{u_1\}, \{u_2\}, \ldots, \{u_n\}$
Nonproportional damping $[C][M]^{-1}[K] \neq [K][M]^{-1}[C]$	$2n$ complex eigenvalues $\lambda_k, k = 1, 2, \ldots, 2n$	$2n$ complex eigenvectors that are nonorthogonal $\{v_k\}, k = 1, 2, \ldots, 2n$

satisfy condition (14.17). The kth eigenvector $\{u_k\}$ of such a beam is governed by

$$\omega_k^2 \{u_k\} = ([\Omega_b] + [K_c])\{u_k\} \tag{14.22}$$

and the related eigenvalues are

$$\lambda_{\pm k} = -\frac{d_v}{2\rho} \pm j\sqrt{\omega_k^2 - \left(\frac{d_v}{2\rho}\right)^2}, \quad j = \sqrt{-1} \tag{14.23}$$

Case 3. Nonproportionally damped system When condition (14.17) is not met, the constrained beam is nonproportionally damped. In this case, Eq. (14.10) has $2n$ complex eigenvalues λ_k and $2n$ complex eigenvectors $\{v_k\}$, $k = 1, 2, \ldots, 2n$. Unlike those of an undamped system, the eigenvectors $\{v_k\}$ of a nonproportionally-damped system do not enjoy the orthogonality relations (14.16). A beam with attached masses and dampers often does not satisfy Eq. (14.17), and is nonproportionally damped.

The above discussion of the eigensolutions of a constrained beam is summarized in Table 14.2.2. Also, refer to Chapter 11 for the effects of damping on the eigensolutions of multiple-degree-of-freedom systems.

14.2.2 System Setup and Eigensolutions

The Toolbox of this chapter provides a function `setbeam` for setting up a constrained beam and computing its eigensolutions; see Window 2.1.

Window 2.1. Function setbeam

MATLAB Function: **setbeam**

Purpose:
To set up a constrained beam and to compute its eigensolutions.

Synopsis:

```
setbeam(rou, EI, L, Dmp_Spec, BC_Spec, SMD_Spec, n_mode)
```

Window 2.1. Function setbeam (*Continued*)

Description:
`setbeam(rou, EI, L, Dmp_Spec, BC_Spec, SMD_Spec, n_mode)` undertakes the following tasks:

(a) It assigns linear density `rou`, bending stiffness `EI`, and length `L` of the beam;
(b) It specifies the damping of the beam by a vector `Dmp_Spec`;
(c) It sets up the beam boundary conditions by a vector `BC_Spec` (see Table 14.2.3);
(d) It specifies the attached SMDs by a matrix `SMD_Spec`;
(e) It sets up the number `n_mode` of terms in the modal series (14.5); and
(f) It computes the eigensolutions of the constrained beam.

By default, `n_mode = 8`. If the beam has no damping, simply replace `Dmp_Spec` by 0 or [ ]. Also, `SMD_Spec` can be replaced by 0 or [ ] if no SMD is attached to the beam. For example,

```
setbeam(rou, EI, L, Dmp_Spec, BC_Spec)
```

sets up a beam without attached SMDs.

Note:
Function `setbeam` must be called before any other functions for dynamic analysis of constrained beams can be used.

According to Window 2.1, a constrained beam is specified by `setbeam` with the following arguments:

(i) The damping in the beam is specified by a vector `Dmp_Spec` in the following options:

(a) Viscous damping. `Dmp_Spec = dv` where `dv` is a viscous damping coefficient.
(b) Modal damping. `Dmp_Spec` is a vector of the form

```
Dmp_Spec = [1 zeta_1, zeta_2 ...zeta_m ]
```

where `zeta_1, zeta_2, ..., zeta_m` are modal damping ratios, i.e., $\xi_1, \xi_2, \ldots \xi_m$. If m is less than the number `n` of terms in Eq. (14.5), the modes of numbers $m+1, m+2, \ldots, n$ will have the same damping ratio as `zeta_m`.
(c) No damping. `Dmp_Spec = [] or 0`.

(ii) The boundary conditions of the beam are specified by a row vector

```
BC_Spec = [BC_Left    BC_Right]
```

where `BC_Left` and `BC_Right` are two subvectors describing the boundary conditions at the left and right ends of the beam respectively; see Table 14.2.3.

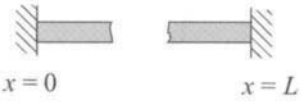
TABLE 14.2.3 Specification of Boundary Conditions

BC_Spec = [BC_Left BC_Right]

(B1) Pinned or hinged end

BC_Left = 1; BC_Right = 1

(B2) Clamped or fixed end



BC_Left = 2; BC_Right = 2

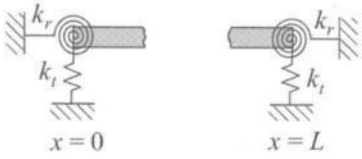
(B3) Free end

BC_Left = 3; BC_Right = 3

(B4) Sliding or guided end

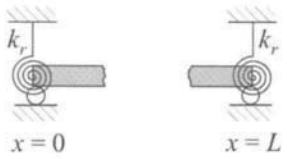
BC_Left = 4; BC_Right = 4

(B5) Elastic-support



BC_Left = [5 k_r k_t]; BC_Right = [5 k_r k_t]

(B6) Hinged-elastic end



BC_Left = [6 k_r]; BC_Right = [6 k_r]

(B7) Sliding-elastic end

BC_Left = [7 k_t]; BC_Right = [7 k_t]

k_r - torsional spring coefficient

k_t - translational spring coefficient

(iii) The locations and parameters of attached SMDs are specified by a matrix

$$\text{SMD\_Spec} = \begin{bmatrix} x_1 & k_1 & m_1 & c_1 \\ x_2 & k_2 & m_2 & c_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_p & k_p & m_p & c_p \end{bmatrix}$$

where the jth row of the matrix specifies the jth SMD set located at $x = x_j$. If one of the elements does not exist, simply set the coefficient of that element to zero.

EXAMPLE 2.1

For the two beam configurations in Fig. 14.2.2, the boundary specification vector is as follows:

Case (a) Clamped-free beam (B2-B3):

```
BC_Spec = [2 3]
```

Case (b) Clamped — sliding-elastic (B2-B7):

```
BC_Spec = [1 7 kt]
```

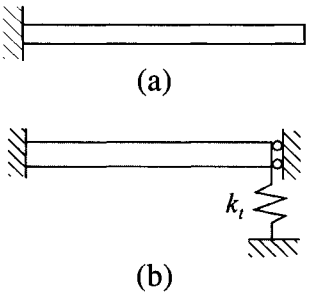


FIGURE 14.2.2

EXAMPLE 2.2

Consider a simply-supported undamped beam with a lumped mass and spring support at $x = a$; see in Fig. 14.2.3. Let the system parameters be $\rho = 16$, $EI = 1$, $L = 2$, $a = 0.7$, $k = 5$, and $m = 2$. The vectors for specification of the beam are

```
Dmp_Spec = 0,     BC_Spec = [1 1]     SMD_Spec = [0.7 5 2 0]
```

The structure is set up by the following commands:

```
>> rou = 16; EI = 1; L = 2; a = 0.7; k = 5; m = 2;
>> Dmp_Spec = 0; BC_Spec = [1 1]; SMD_Spec = [a k m 0];
>> setbeam(rou, EI, L, Dmp_Spec, BC_Spec, SMD_Spec)
```

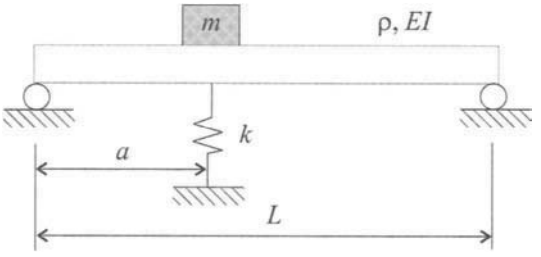


FIGURE 14.2.3

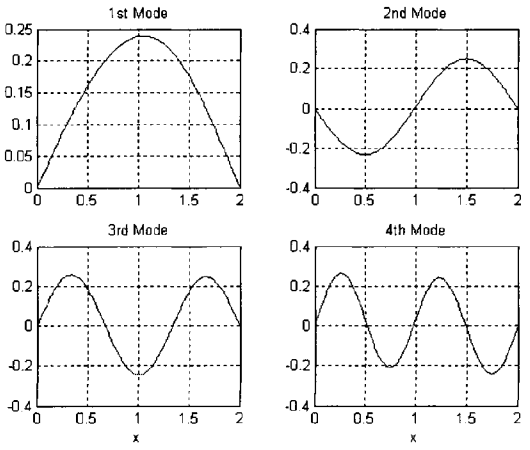


FIGURE 14.2.4

which produce the following results and the first four mode shapes plotted in Fig. 14.2.4.

```
Dynamic Analysis of A Constrained Beam
======================================
Beam parameters:
  Linear density = 16
  Bending stiffness = 1
  Length = 2

Boundary conditions of beam:
  Left end: simply-supported
  Right end: simply-supported

Damping of beam:
  The beam has no damping

Attached springs, masses, and dampers:
  at x = 0.7: k = 5, m = 2, c = 0

Number of modes selected for simulation: 8
Eigenvalues of the Constrained Beam
-----------------------------------
The system is undamped with natural frequencies
  mode 1: omega = 0.75262 rad/s
  mode 2: omega = 2.4145 rad/s
  mode 3: omega = 5.5447 rad/s
  mode 4: omega = 9.4231 rad/s
  mode 5: omega = 15.0733 rad/s
  mode 6: omega = 22.1049 rad/s
  mode 7: omega = 28.974 rad/s
  mode 8: omega = 38.9876 rad/s
```

To obtain the information of the constrained beam set up by `setbeam`, function **`systinfo`** can be called as follows:

```
>> systinfo
```

After function `setbeam` is executed, functions `plotmsh` and `animode` can be called to display and animate the modes of vibration of the beam structure; see Windows 2.2 and 2.3.

Window 2.2. Function `plotmsh`

MATLAB Function: **`plotmsh`**

Purpose:
To plot the mode shapes of a constrained or combined beam.

Synopsis:

```
plotmsh(k)
plotmsh(k, npts)
[y, x] = plotmsh(k, npts)
```

Description:
`plotmsh(k)` plots the *k*th mode shape of a constrained or combined beam over its length.

`plotmsh(k, npts)` plots the *k*th mode shape of a constrained or combined beam at `npts` equally-spaced points over its length. By default, `npts = 201`.

`[y, x] = plotmsh(k, npts)` returns the computed data on the *k*th mode shape in a vector `y` and spatial points of computation in a vector `x`. With `x` and `y`, the *k*th mode shape can be plotted later on by the MATLAB function `plot(x, y)`.

Window 2.3. Function `animode`

MATLAB Function: **`animode`**

Purpose:
To animate a mode of vibration of a constrained or combined beam.

Synopsis:

```
animode(k)
animode(k, IS_control, tf, n_frames)
F = animode(k, IS_control, tf, n_frames)
```

Description:
`animode(k)` animates the *k*th mode of vibration of the beam structure.

`animode(k, IS_control, tf, n_frames)` plays animated vibration of the beam in a specified manner, where `IS_control` is an animation control parameter with the options:

```
IS_control = 0    play frames continuously
IS_control = 1    play frames one by one
```

`tf` is total animation time, and `n_frames` is the number of frames in the animation. By default, `IS_control = 0`, `tf = 8*T` with T being the period of the mode, and `n_frames = 97`. Any of `IS_control`, `tf`, and `n_frames` can be replaced by [ ] as a placeholder for its default setup.

`F = animode(k, IS_control, tf, n_frames)` returns a set of frames of beam vibration in a matrix `F`, which can be played later on by the MATLAB function `movie(F)`.

EXAMPLE 2.3

The constrained beam in Fig. 14.2.5 has the following parameters:

Beam (undamped): $\rho = 4$, $EI = 120$, $L = 3$
SMDs: at $x_1 = 1$, $k = 10$; at $x_2 = 2$, $m = 2$, $c = 5$

By

```
>> rou = 4; EI = 120; L = 3; Dmp_Spec = 0; BC_Spec = [2 1];
>> SMD_Spec = [1 10 0 0; 2 0 2 5];
>> setbeam(rou, EI, L, Dmp_Spec, BC_Spec, SMD_Spec)
>> plotmsh(1)
```

the spatial distribution of the first mode shape (eigenfunction) of the beam is plotted in Fig. 14.2.6. The eigenfunction has both real and imaginary parts because the beam structure has nonproportional damping.

To animate the first mode of vibration, type

```
>> animode(1, 1, 0.7748, 6)
```

which yields the six frames in Fig. 14.2.7. Here, the total simulation time is equal to the first damped period, namely, $\texttt{tf} = 2\pi/\omega_{d1} = 2\pi/8.1097 = 0.7748$.

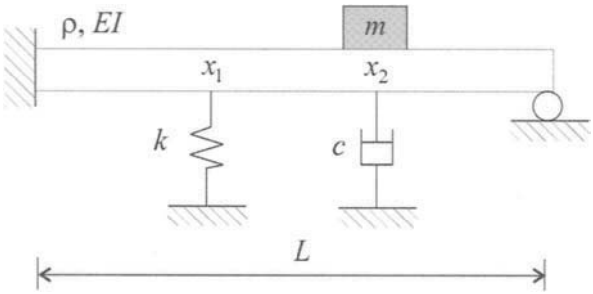


FIGURE 14.2.5

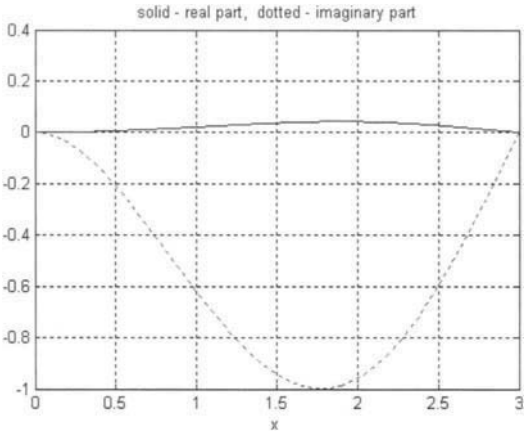


FIGURE 14.2.6 The real and imaginary parts of the first mode shape.

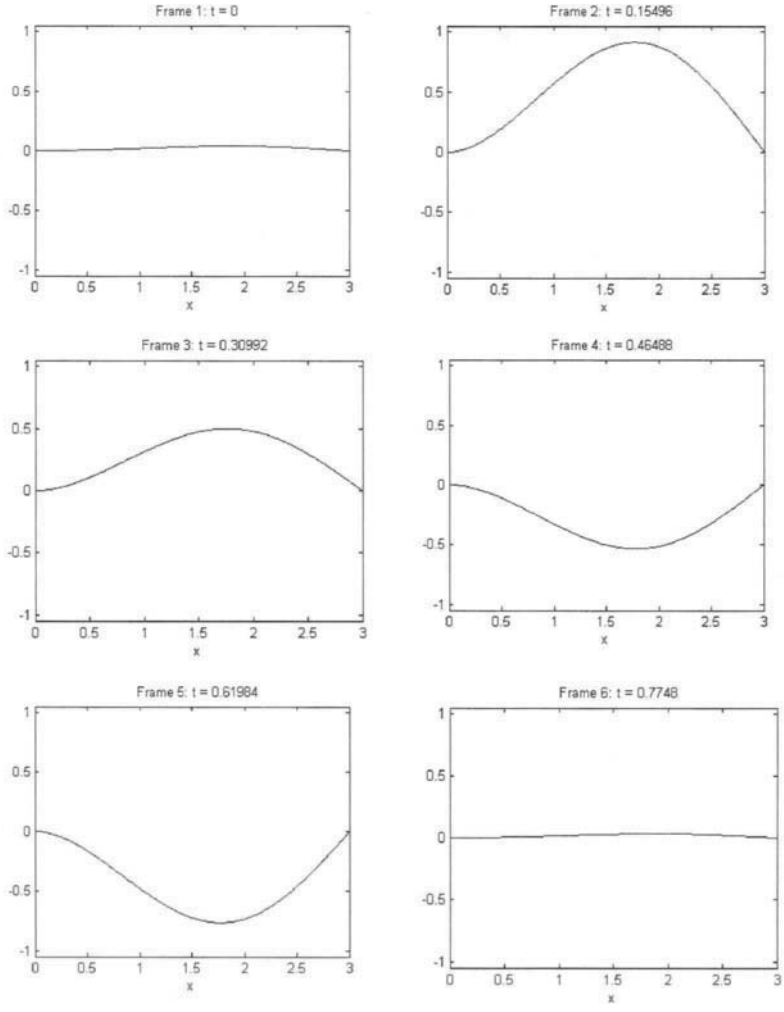


FIGURE 14.2.7 Frames of the animated beam vibration in the first mode.

14.2.3 Eigenvalue Locus

In the design of a beam structure, it is often of interest to examine the variation of its eigenvalues subject to the change in a parameter of an attached SMD set. For instance, the natural frequencies of the beam structure in Fig. 14.2.3 vary when the spring coefficient k changes. In other words, the natural frequencies are functions of k. The curve of an eigenvalue plotted against a parameter of the SMD is called the *eigenvalue locus* of the beam structure.

Three cases of eigenvalue loci of a constrained beam are considered here:

(a) For the beam structure with a newly added SMD set, its eigenvalue loci versus the SMD location x_a are obtained; see Fig. 14.2.8. The parameters k, m, and c of the added SMD are fixed.
(b) For the beam structure with a newly added element, its eigenvalue loci versus the element coefficient (k, m, or c) are obtained; see Fig. 14.2.9a. The element location x_a is fixed.
(c) For the beam structure, its eigenvalue loci versus one of the parameters (k, m, or c) of an existing SMD set are obtained; see Fig. 14.2.9b.

The Toolbox of this chapter offers a function `eiglocus` for plotting eigenvalue loci of constrained beams; see Window 2.4.

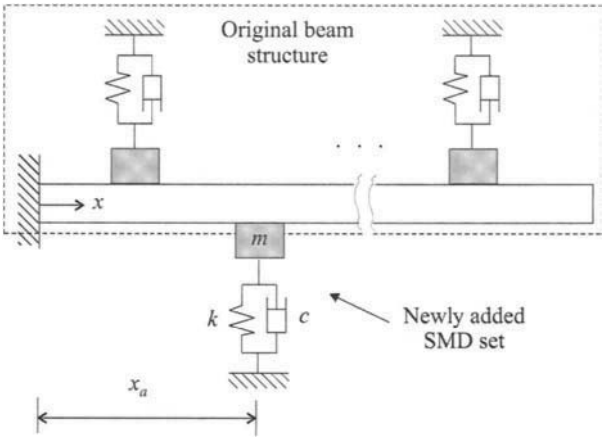


FIGURE 14.2.8 A constrained beam with a newly added SMD of variable location x_a.

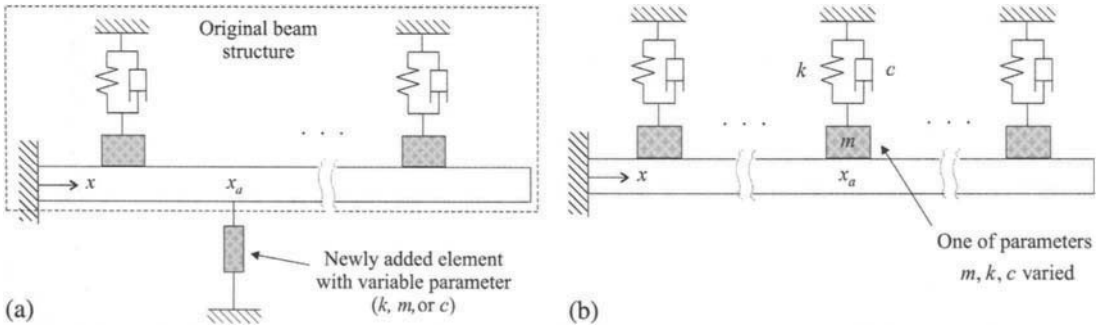


FIGURE 14.2.9 A constrained beam with (a) a newly added element of variable coefficient, or (b) an existing SMD set of one variable coefficient.

Window 2.4. Function `eiglocus`

MATLAB Function: **`eiglocus`**

Purpose:
To plot the eigenvalue loci of a constrained beam.

Synopsis:

```
eiglocus(Locus_Spec, p0, pf, npts)
[y, p] = eiglocus(Locus_Spec, p0, pf, npts)
```

Description:
`eiglocus(Locus_Spec, p0, pf, npts)` plots the loci of the first three eigenvalues of the beam structure. `Locus_Spec` is a vector specifying the type of eigenvalue loci in the following options:

`Locus_Spec = [0 k m c]` for eigenvalue loci versus the location of an added SMD set of coefficients `k`, `m`, and `c` (Fig. 14.2.8);

`Locus_Spec = [1 xp]` for eigenvalue loci versus the coefficient of an added spring at location `xp` (Fig. 14.2.9a);

`Locus_Spec = [2 xp]` for eigenvalue loci versus the coefficient of an added mass at location `xp` (Fig. 14.2.9a);

`Locus_Spec = [3 xp]` for eigenvalue loci versus the coefficient of an added damper at location `xp` (Fig. 14.2.9a);

`Locus_Spec = [11 xp]` for eigenvalue loci versus the spring coefficient of an existing SMD at location `xp` (Fig. 14.2.9b);

`Locus_Spec = [12 xp]` for eigenvalue loci versus the mass of an existing SMD at location `xp` (Fig. 14.2.9b); and

`Locus_Spec = [13 xp]` for eigenvalue loci versus the damping coefficient of an existing SMD at location `xp` (Fig. 14.2.9b).

`p0` and `pf` are the initial and final values of the variable parameter against which the eigenvalue loci are plotted. For example, if `Locus_Spec = [1 xp]`, the coefficient `k` of the added spring varies in the range $p0 \leq k \leq pf$. The integer `npts` is the number of equally-spaced points in the region `[p0, pf]` selected for plotting the loci. By default, `npts = 201`.

`[y, p] = eiglocus(Locus_Spec, p0, pf, npts)` returns the eigenvalue loci in an n-row matrix `y`, and the values of the variable parameter (location or element coefficient) in a row vector `p`. Here n is the total number of modes of the constrained beam, as indicated by Eq. (14.5). With `p` and `y`, the eigenvalue loci can be plotted by the MATLAB function `plot` as below:

(i) For an undamped beam, `plot(p, y(k,:))` plots the locus of the kth natural frequency;
(ii) For a damped beam, `plot(p, real(y(k,:)))` and `plot(p, imag(y(k,:)))` plot the loci of the real and imaginary parts of the kth eigenvalue, respectively.

EXAMPLE 2.4

In Fig. 14.2.10, a clamped-pinned beam with a mounted lumped mass has the following parameters: $\rho = 0.5$, $EI = 8$, $L = 2$, $x_a = 1.3$. The loci of the first three natural frequencies of the beam structure versus m (from 0 to 20) are plotted in Fig 14.2.11 by the following commands:

```
>> rou = 0.5; EI = 8; L = 2;
>> setbeam(rou, EI, L, 0, [2 1])
>> eiglocus([2 1.3], 0, 20)
```

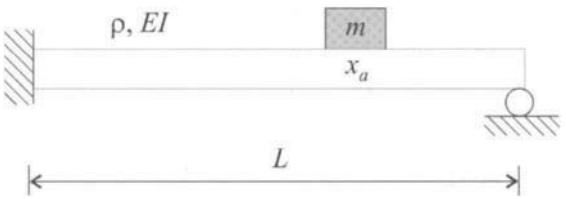


FIGURE 14.2.10

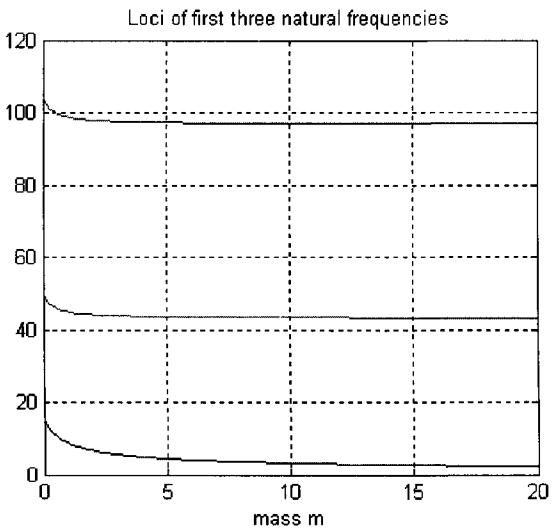


FIGURE 14.2.11

EXAMPLE 2.5

In Fig. 14.2.12, a clamped-guided beam with viscous damping is connected to a mass and a damper at x_1. Let the parameters of the structure be $\rho = 1$, $EI = 60$, $d_v = 2.5$, $L = 4$, $x_1 = 2.5$, $m = 0.75$, $c = 0.9$. Now, attach a spring of coefficient $k = 50$ to the beam. Let the spring location x_a vary from 0 to L. (The spring moves from the left end

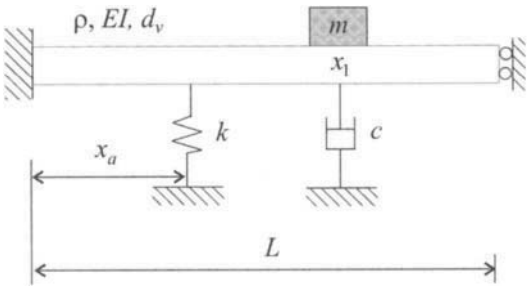


FIGURE 14.2.12

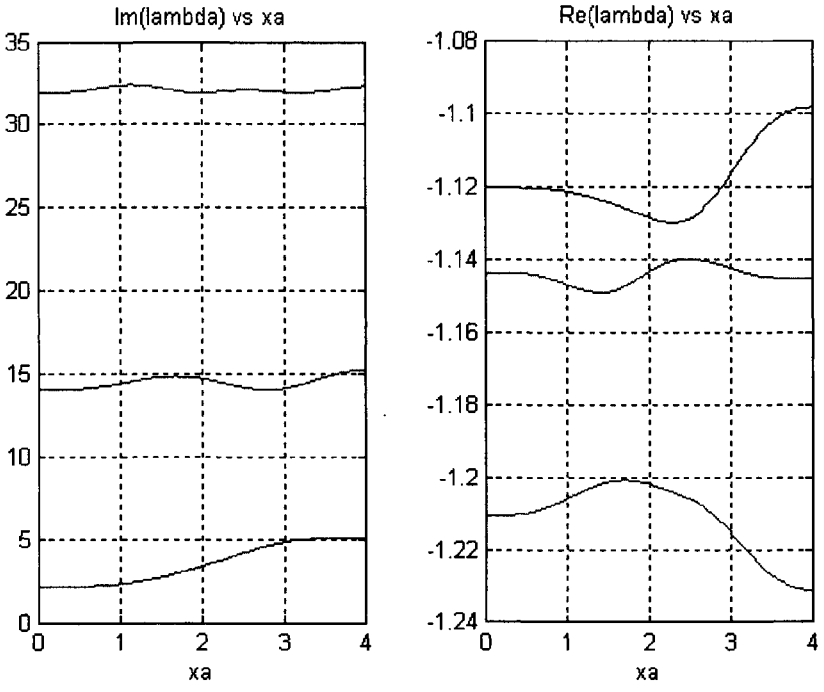


FIGURE 14.2.13

to the right end of the beam.) The loci of the imaginary and real imaginary parts (Re λ_k, Im λ_k) of the first three eigenvalues of the structure versus x_a are plotted in Fig. 14.2.13 by the following commands:

```
>> rou = 1; EI = 60; L = 4;
>> SMD_Spec = [2.5 0 0.75 0.9];
>> setbeam(rou, EI, L, 2.5, [2 4], SMD_Spec)
>> [y, p] = eiglocus([0 50 0 0], 0, L);
>> subplot(1,2,1); plot(p, imag(y(1,:)), p, imag(y(2,:)),
   p, imag(y(3,:)))
>> title('Im(lambda) vs xa'); grid; xlabel('xa')
>> subplot(1,2,2); plot(p, real(y(1,:)), p, real(y(2,:)),
   p, real(y(3,:)))
>> title('Re(lambda) vs xa'); grid; xlabel('xa')
```

14.2.4 Frequency Response

Consider a constrained beam subject to a pointwise sinusoidal force of amplitude f_0 that is located at $x = x_f$. Equation (14.1) becomes

$$\rho \frac{\partial^2}{\partial t^2} w(x,t) + d_v \frac{\partial}{\partial t} w(x,t) + EI \frac{\partial^4}{\partial x^4} w(x,t) = f_0 \, e^{j\omega t} \, \delta(x - x_f) - f_c(x,t) \tag{14.24}$$

where $j = \sqrt{-1}$ and ω is the excitation frequency. The steady-state solution of the above, by the modal series (14.5), is expressed as

$$w_{ss}(x,t) = [\Phi(x)]^T \{q(t)\} \tag{14.25}$$

where the vector of generalized coordinates is of the form

$$\{q(t)\} = \{Q\} \, e^{j\omega t} \tag{14.26}$$

Substitute Eq. (14.26) into Eq. (14.24) and use Eq. (14.8), to obtain

$$\left(-\omega^2 [M] + j\omega [D] + [K]\right) \{Q\} = f_0 \left[\Phi(x_f)\right] \tag{14.27}$$

which leads to

$$\{Q\} = f_0 \left(-\omega^2 [M] + j\omega [D] + [K]\right)^{-1} \left[\Phi(x_f)\right] \tag{14.28}$$

Thus, the steady-state displacement of the beam structure is

$$w_{ss}(x,t) = W(x,\omega) \, e^{j\omega t} \tag{14.29}$$

where

$$W(x,\omega) = f_0 \, [\Phi(x)]^T \left(-\omega^2 [M] + j\omega [D] + [K]\right)^{-1} \left[\Phi(x_f)\right] \tag{14.30}$$

The $W(x,\omega)$ is called the *frequency response* of the beam structure. Equation (14.29) can also be written as

$$w_{ss}(x,t) = H \, e^{j(\omega t + \theta)} \tag{14.31}$$

where H and θ are the magnitude and phase of the frequency response

$$H = |W(x,\omega)|, \quad \theta = \angle W(x,\omega) \tag{14.32}$$

The above result is obtained based on a sinusoidal force of complex format ($e^{j\omega t}$). The frequency response $W(x,\omega)$ is applicable to the case of real forces. For instance, the steady-state response of the system under a sinusoidal force $f_0 \sin(\omega t + \theta_0)$ is

$$w_{ss}(x,t) = H \sin(\omega t + \theta_0 + \theta) \tag{14.33}$$

with the magnitude H and θ given in Eq. (14.32).

The Toolbox of this chapter provides a function `frfbeam` for plotting the frequency response of a constrained beam; see Window 2.5.

Window 2.5. Function frfbeam

MATLAB Function: **frfbeam**

Purpose:
To plot the magnitude and phase of steady-state displacement of a constrained beam subject to a pointwise sinusoidal load.

Synopsis:

```
frfbeam(xr, xf, f0)
frfbeam(xr, xf, f0, omg0, omgf, npts)
[y, freq] = frfbeam(xr, xf, f0, omg0, omgf, npts)
```

Description:
`frfbeam(xr, xf, f0)` plots the magnitude and phase of steady-state displacement of a constrained beam at `xr` versus the frequency of a sinusoidal force that is applied to the beam at `xf`, where `f0` is the amplitude of the force. Also, `frfbeam(xr, xf)` plots the beam frequency response for `f0 = 1`.

`frfbeam(xr, xf, f0, omg0, omgf, npts)` plots the steady-state displacement of a constrained beam in a specified frequency region `[omg0, omgf]`, where `omg0` and `omgf` are the lower and upper bounds of the frequency region, and `npts` is the number of points in the region for simulation. By default, `omg0 = 0`, `omgf` is large than the third natural or damped frequency of the beam, and `npts = 1001`.

`[y, freq] = frfbeam(xr, xf, f0, omg0, omgf, npts)` returns the computed frequency response in a matrix `y` and frequencies of simulation in a vector `freq`. With `y` and `freq`, the frequency response can be plotted as follows:

for magnitude	`plot(freq, y(1,: ))`
for phase	`plot(freq, y(2,: ))`

EXAMPLE 2.6

Consider the constrained beam in Example 2.3. A sinusoidal force of amplitude $f_0 = 12$ is applied at $x_f = 1.5$. Assume that each mode of the beam has 2% modal damping. By

```
>> rou = 4; EI = 120; L = 3;
>> Dmp = [1 0.02]; BC = [2 1];
>> SMD = [1 10 0 0; 2 0 2 5];
>> setbeam(rou, EI, L, Dmp, BC, SMD)
>> f0 = 12; xf = 1.5; xr = 2;
>> frfbeam(xr, xf, f0)
```

the frequency response of the beam at the mass location ($x = 2$) is plotted in Fig. 14.2.14.

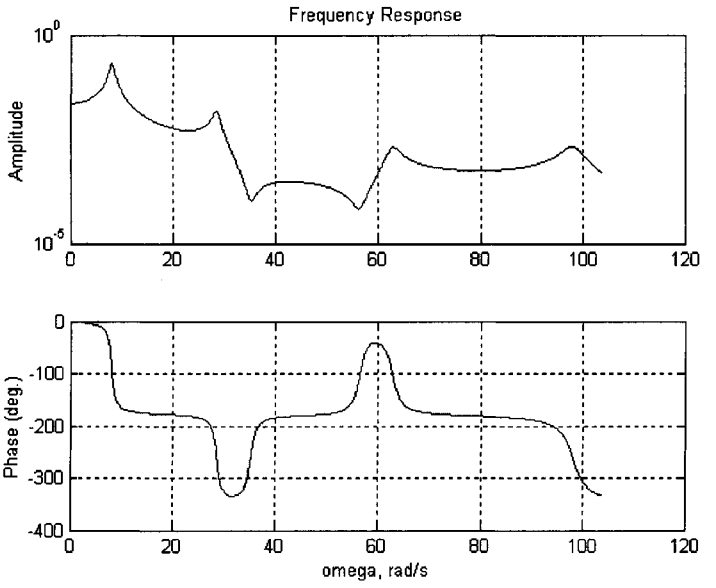


FIGURE 14.2.14

14.2.5 Multispan Beam Structures

When the coefficient of an attached spring goes to infinite, that spring behaves like a hinge or pin. See Fig. 14.2.15, for example, where an elastically supported beam becomes a two-span beam structure as the spring coefficient k goes infinite. This feature can be used in modeling and analysis of multispan beam structures. For instance, in system setup by function `setbeam`, a hinge can be described by a spring of a large coefficient $k = \mu \cdot EI/L^3$, where EI and L are the bending stiffness and length of the beam, and μ is a large number (say $\mu \geq 10^6$).

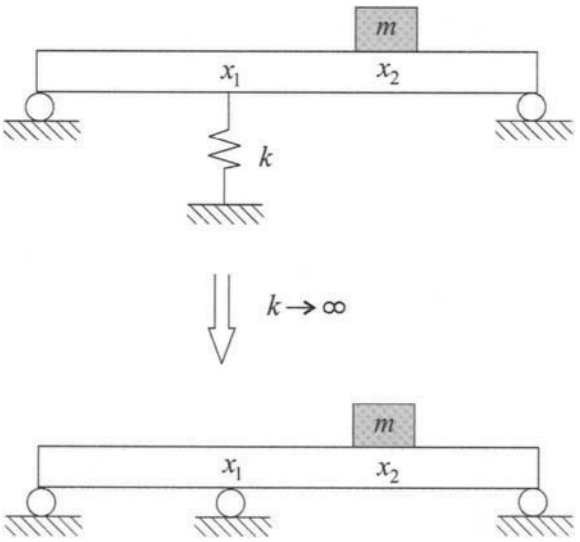


FIGURE 14.2.15

With the above treatment, functions `plotmsh`, `animode`, `eiglocus`, and `frfbeam` can be directly called to compute and display the dynamic response of multispan beam structures. Also, function `setbeam3` in Section 14.4 can be used to compute the eigensolutions of an undamped multispan beam.

EXAMPLE 2.7

Consider the two-span beam in Fig. 14.2.15 with the parameters $\rho = 0.3$, $EI = 40$, $L = 2, x_1 = 1.0, x_2 = 1.5, m = 6$. The hinge at x_1 is modeled as a spring of coefficient $k = 10^7 \cdot \frac{EI}{L^3} = 5 \times 10^7$. By the commands

```
>> rou = 0.3; EI = 40; L = 2; BC = [1 1];
>> k = 1e7*EI/L^3;
>> SMD = [1 k 0 0; 1.5 0 6 0];
>> setbeam(rou, EI, L, 0, BC, SMD, 20)
```

the natural frequencies of the two-span beam are obtained below:

```
Eigenvalues of the Beam with Attached SMDs
------------------------------------------
The system is undamped with natural frequencies:
  mode 1: omega = 20.8167 rad/sec
  mode 2: omega = 148.7942 rad/sec
  mode 3: omega = 455.8575 rad/sec
  mode 4: omega = 548.8144 rad/sec
  mode 5: omega = 804.2443 rad/sec
  mode 6: omega = 1139.33 rad/sec
  mode 7: omega = 1823.43 rad/sec
  mode 8: omega = 2011.9409 rad/sec
```

and the first four mode shapes are plotted in Fig. 14.2.16.

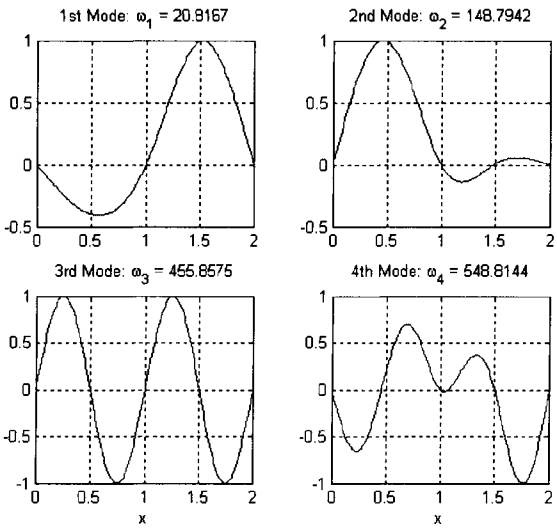


FIGURE 14.2.16

14.2.6 Transient Response

The transient response of constrained beams subject to external and initial disturbances is discussed in Section 14.3.3, where a beam with attached SMDs is a special case of combined beams. Functions `trbeam2`, `trbeam2s`, and `beammovie2` are available from the Toolbox for computing and animating the transient response of a constrained beam.

14.3 Combined Beams

Combined beams are commonly used models of structures in various engineering applications, including automobile suspension systems, airplanes with mounted engines and fuselages, and buildings and bridges with mass dampers. A typical combined beam is shown in Fig. 14.3.1, where a uniform Euler-Bernoulli beam is constrained by p sets of springs, lumped masses, and dampers (SMD), is combined with q one-degree-of-freedom oscillators, and carries s elastically-mounted rigid bodies.

The Toolbox of this chapter offers several MATLAB functions for dynamic analysis of combined beams. The solution procedure by the Toolbox takes the following two steps:

Step 1. Set up system parameters and boundary conditions

To set up a combined beam, call function `setbeam2`. Function `setbeam2` also yields the eigensolutions of the beam.

Step 2. Compute and display system response

To plot mode shapes, call function `plotmsh`.

To animate modes of vibration, call function `animode`.

To plot transient response, call function `trbeam2`.

To compute frequency response, call function `frfbeam2`.

These functions, along with others from the Toolbox, will be introduced in sequel.

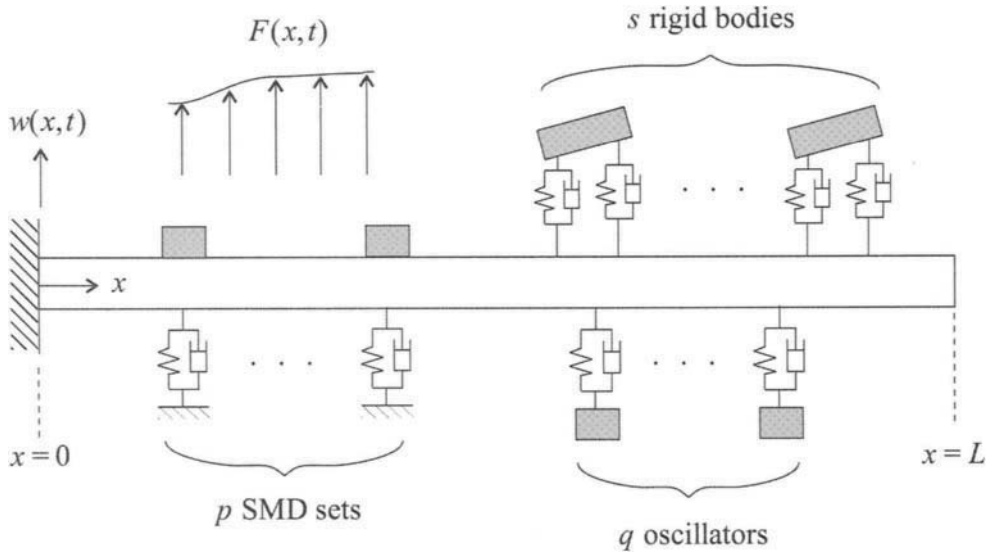


FIGURE 14.3.1 Schematic of a combined beam.

14.3.1 System Description

Governing Equations Figure 14.3.2 shows an SMD set, an oscillator, and a rigid body that are combined to a beam structure. The SMD set in Fig. 14.3.2a has been fully characterized in Sections 14.2.1 and 14.2.2. In Fig. 14.3.2b, the oscillator is a mass $\hat{m}_j$ connected to the beam at location ξ_j by a spring of coefficient $\hat{k}_j$ and a damper of coefficient $\hat{c}_j$. The oscillator has one degree of freedom with a vertical displacement $y_j(t)$. In Fig. 14.3.2c, the rigid body of mass M_j and moment of inertia I_j is mounted to the beam at two points with supporting springs and dampers:

at the left point x_{Lj}: spring k_{Lj}, damper c_{Lj}
at the right point x_{Rj}: spring k_{Rj}, damper c_{Rj}

The rigid body has two degrees of freedom, with a vertical displacement $z_j(t)$ and a rotation $\theta_j(t)$ with respect to its center of mass, which is a distance b_j from the left supporting point.

With the above description, the transverse displacement $w(x,t)$ of the combined beam shown in Fig. 14.3.1 is governed by

$$\rho \frac{\partial^2}{\partial t^2} w(x,t) + d_v \frac{\partial}{\partial t} w(x,t) + EI \frac{\partial^4}{\partial x^4} w(x,t) = F(x,t) - F_c(x,t), \quad 0 \le x \le L \tag{14.34}$$

and appropriate boundary and initial conditions, where ρ, EI, and L are the linear density (mass per unit length), bending stiffness, and length of the beam, respectively; d_v is a viscous damping coefficient; $F(x,t)$ is an external force; and $F_c(x,t)$ is the resultant constraint force due to the attached SMDs, oscillators, and rigid bodies. Besides viscous damping, modal damping can also be introduced. The external force considered here is of the form

$$F(x,t) = b(x) \cdot f(t) \tag{14.35}$$

where function $b(x)$ describes the spatial distribution of the force. The resultant constraint force $F_c(x,t)$ is written as

$$\begin{aligned} F_c(x,t) = &\sum_{i=1}^{p} f_{a,i}(t)\delta(x - x_i) + \sum_{j=1}^{q} f_{o,j}(t)\delta(x - \xi_j) \\ &+ \sum_{k=1}^{s} \left(f_{L,k}(t)\delta(x - x_{Lk}) + f_{R,k}(t)\delta(x - x_{Rk}) \right) \end{aligned} \tag{14.36}$$

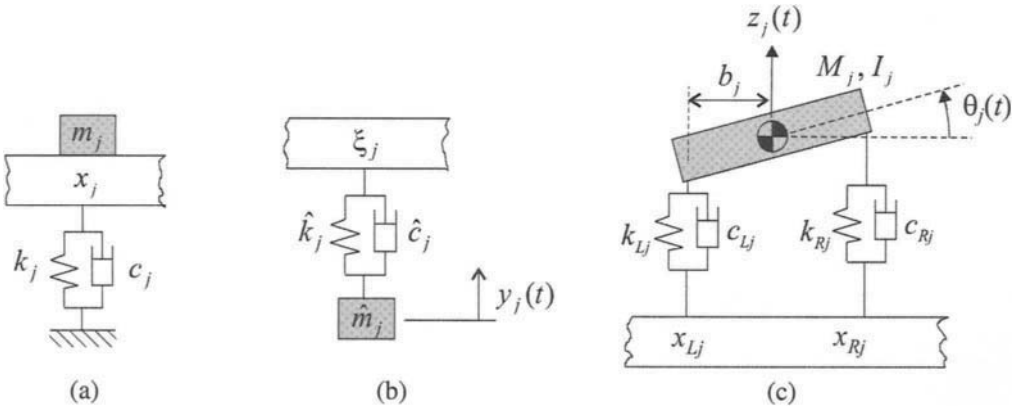


FIGURE 14.3.2 Components of a combined beam: (a) SMD set, (b) oscillator, and (c) elastically-mounted rigid body.

where

$$
\begin{aligned}
f_{a,i}(t) &= m_i w_{,tt}(x_i,t) + c_i w_{,t}(x_i,t) + k_i w(x_i,t) \\
f_{o,j}(t) &= \hat{c}_j \left(w_{,t}(x_j,t) - \dot{y}_j(t)\right) + \hat{k}_j \left(w(x_j,t) - y_j(t)\right) \\
f_{L,k}(t) &= c_{Lk} \left(w_{,t}(x_{Lj},t) - (\dot{z}_k(t) - b_j\dot{\theta}_j(t))\right) \\
&\quad + k_{Lk} \left(w(x_{Lj},t) - (z_k(t) - b_j\theta_j(t))\right) \\
f_{R,k}(t) &= c_{Rk} \left(w_{,t}(x_{Rj},t) - (\dot{z}_k(t) + (a_j - b_j)\dot{\theta}_j(t))\right) \\
&\quad + k_{Rk} \left(w(x_{Rj},t) - (z_k(t) + (a_j - b_j)\theta_j(t))\right)
\end{aligned}
\tag{14.37}
$$

$a_j = x_{Rj} - x_{Lj}$, $\delta(\,)$ is the delta function, and $w_{,t} = \partial w/\partial t$. As can be seen, $f_{a,i}(t)$ is the constraint force applied to the beam by the ith SMD set; $f_{o,j}(t)$ is the constraint force by the jth oscillator; and $f_{L,k}(t)$ and $f_{R,k}(t)$ are the constraint forces by the kth rigid body. The boundary conditions of the constrained beam are given in Table 14.2.1.

The motion of the jth oscillator, by Fig. 14.3.2b, is governed by

$$
\hat{m}_j \ddot{y}_j(t) = f_{o,j}(t) \tag{14.38}
$$

and the motion of the jth rigid body, by Fig. 14.3.2c, is described by

$$
\begin{aligned}
M_j \ddot{z}_j(t) &= f_{L,j}(t) + f_{R,j}(t) \\
I_j \ddot{\theta}_j(t) &= -b_j f_{L,j}(t) + (a_j - b_j) f_{R,j}(t)
\end{aligned}
\tag{14.39}
$$

where the constraint forces are given by Eqs. (14.37). The positive direction of the translational displacements y_j and z_j are upward; the positive direction of the rotational displacement θ_j is counterclockwise.

Modal Expansion The transverse displacement $w(x,t)$ of the beam is approximated by an n-term modal series

$$
w(x,t) \approx \sum_{k=1}^{n} \phi_k(x) q_k(t) = [\Phi(x)]^T \{q(t)\} \tag{14.40}
$$

which is the same as given in Eq. (14.5).

Substitute the modal series (14.40) into Eq. (14.34) and apply the orthonormal conditions (14.7), to obtain the matrix differential equation for the combined beam

$$
[M]\{\ddot{q}_a(t)\} + [D]\{\dot{q}_a(t)\} + [K]\{q_a(t)\} = \{B_a\} f(t) \tag{14.41}
$$

where the vector of displacements

$$
\{q_a(t)\} = \begin{Bmatrix} \{q(t)\} \\ \{z_O(t)\} \\ \{z_R(t)\} \end{Bmatrix} \tag{14.42}
$$

with

$$
\{z_O(t)\} = \begin{pmatrix} y_1(t) & \cdots & y_q(t) \end{pmatrix}^T, \quad \{z_R(t)\} = \begin{pmatrix} z_1(t) & \theta_1(t) & \cdots & z_s(t) & \theta_s(t) \end{pmatrix}^T
$$

which are the displacement vectors of the oscillators and rigid bodies, respectively; and the force distribution vector

$$\{B_a\} = \begin{pmatrix} \{B_f\} \\ \{0\}_q \\ \{0\}_{2s} \end{pmatrix} \tag{14.43}$$

with

$$\{B_f\} = \int_0^L [\Phi(x)]\, b(x) dx$$

and $\{0\}_q$ and $\{0\}_{2s}$ being zero vectors of order q and $2s$, respectively. The mass, damping, and stiffness matrices in Eq. (14.41) are given by

$$[M] = \begin{bmatrix} [I] + [M_c] & 0 & 0 \\ 0 & \left[\hat{M}_O\right] & 0 \\ 0 & 0 & \left[\hat{M}_R\right] \end{bmatrix}$$

$$[D] = \begin{bmatrix} [D_b] + [D_c] + [D_O] + [D_R] & -[D_{bO}] & -[D_{bR}] \\ -[D_{Ob}] & \left[\hat{D}_O\right] & 0 \\ -[D_{Rb}] & 0 & \left[\hat{D}_R\right] \end{bmatrix} \tag{14.44}$$

$$[K] = \begin{bmatrix} [\Omega_b] + [K_c] + [K_O] + [K_R] & -[K_{bO}] & -[K_{bR}] \\ -[K_{Ob}] & \left[\hat{K}_O\right] & 0 \\ -[K_{Rb}] & 0 & \left[\hat{K}_R\right] \end{bmatrix}$$

where $[I]$ is an identity matrix of order n; $[M_c]$, $[D_b]$, $[D_c]$, $[\Omega_b]$, and $[K_c]$ have been defined in Eq. (14.9); and

$$[D_O] = \sum_{j=1}^{q} \hat{c}_j \left[\Phi(\xi_j)\right] \left[\Phi(\xi_j)\right]^T, \quad [K_O] = \sum_{j=1}^{q} \hat{k}_j \left[\Phi(\xi_j)\right] \left[\Phi(\xi_j)\right]^T$$

$$[D_R] = \sum_{j=1}^{s} \left(c_{Lj} \left[\Phi(x_{Lj})\right] \left[\Phi(x_{Lj})\right]^T + c_{Rj} \left[\Phi(x_{Rj})\right] \left[\Phi(x_{Rj})\right]^T \right)$$

$$[K_R] = \sum_{j=1}^{s} \left(k_{Lj} \left[\Phi(x_{Lj})\right] \left[\Phi(x_{Lj})\right]^T + k_{Rj} \left[\Phi(x_{Rj})\right] \left[\Phi(x_{Rj})\right]^T \right)$$

$$\left[\hat{M}_O\right] = \underset{1 \le j \le q}{\text{diag}} \left\{\hat{m}_j\right\}, \quad \left[\hat{D}_O\right] = \underset{1 \le j \le q}{\text{diag}} \left\{\hat{c}_j\right\}, \quad \left[\hat{K}_O\right] = \underset{1 \le j \le q}{\text{diag}} \left\{\hat{k}_j\right\}$$

$$\left[\hat{M}_R\right] = \text{diag}\left\{\hat{M}_{R,1} \quad \cdots \quad \hat{M}_{R,q}\right\}, \quad \text{with} \quad \hat{M}_{R,j} = \begin{bmatrix} M_j & 0 \\ 0 & I_j \end{bmatrix} \tag{14.45}$$

$$\left[\hat{D}_R\right] = \text{diag}\left\{\hat{D}_{R,1} \quad \cdots \quad \hat{D}_{R,s}\right\}$$

$$\text{with } \hat{D}_{R,j} = c_{Lj}\begin{bmatrix} 1 & -b_j \\ -b_j & -b_j^2 \end{bmatrix} + c_{Rj}\begin{bmatrix} 1 & a_j - b_j \\ a_j - b_j & (a_j - b_j)^2 \end{bmatrix}$$

$$\left[\hat{K}_R\right] = \text{diag}\left\{\hat{K}_{R,1} \quad \cdots \quad \hat{K}_{R,s}\right\}$$

$$\text{with } \hat{K}_{R,j} = k_{Lj}\begin{bmatrix} 1 & -b_j \\ -b_j & b_j^2 \end{bmatrix} + k_{Rj}\begin{bmatrix} 1 & a_j - b_j \\ a_j - b_j & (a_j - b_j)^2 \end{bmatrix}$$

$$[D_{bO}] = [D_{Ob}]^T = \left[\hat{c}_1\left[\Phi(\xi_1)\right] \quad \hat{c}_2\left[\Phi(\xi_2)\right] \quad \cdots \quad \hat{c}_q\left[\Phi(\xi_q)\right]\right]$$

$$[K_{bO}] = [K_{Ob}]^T = \left[\hat{k}_1\left[\Phi(\xi_1)\right] \quad \hat{k}_2\left[\Phi(\xi_2)\right] \quad \cdots \quad \hat{k}_q\left[\Phi(\xi_q)\right]\right]$$

$$[D_{bR}] = [D_{Rb}]^T = \left[D_{bR,1} \quad D_{bR,2} \quad \cdots \quad D_{bR,s}\right]$$

$$[K_{bR}] = [K_{Rb}]^T = \left[K_{bR,1} \quad K_{bR,2} \quad \cdots \quad K_{bR,s}\right]$$

with

$$D_{bR,j} = c_{Lj}\left[\Phi(x_{Lj})\right]\left[1 \quad -b_j\right] + c_{Rj}\left[\Phi(x_{Rj})\right]\left[1 \quad a_j - b_j\right]$$

$$K_{bR,j} = k_{Lj}\left[\Phi(x_{Lj})\right]\left[1 \quad -b_j\right] + k_{Rj}\left[\Phi(x_{Rj})\right]\left[1 \quad a_j - b_j\right]$$

Equation (14.41) is a discretized model of the combined beam with $n + q + 2s$ degrees of freedom.

Eigenvalue Problem The eigenvalue problem of the combined beam is defined by

$$\left(\lambda_k^2[M] + \lambda_k[D] + [K]\right)\{v_k\} = 0 \tag{14.46}$$

where λ_k is the kth eigenvalue and $\{v_k\}$ is the associate eigenvector. The eigenvector $\{v_k\}$ can be written as

$$\{v_k\} = \begin{pmatrix} \{v_{b,k}\} \\ \{v_{O,k}\} \\ \{v_{R,k}\} \end{pmatrix} \tag{14.47}$$

where $\{v_{b,k}\}$ is an n-vector describing the n coordinates of the beam in the modal expansion (14.40); $\{v_{O,k}\}$ is a q-vector describing the displacements of the q oscillators; and $\{v_{R,k}\}$ is a $2s$-vector describing the displacements of the s rigid bodies. With λ_k and $\{v_{b,k}\}$, the displacement of the beam in the kth mode of vibration is given by

$$w_k(x,t) = \psi_k(x)e^{\lambda_k t}, \quad 0 \le x \le L \tag{14.48}$$

where the mode shape function

$$\psi_k(x) = [\Phi(x)]^T \{v_{b,k}\} \tag{14.49}$$

In addition, the displacements of the oscillators and rigid bodies in the kth mode of vibration can be expressed by $\{v_{O,k}\} e^{\lambda_k t}$ and $\{v_{R,k}\} e^{\lambda_k t}$, respectively.

The format of the eigensolutions depends on the damping status of the combined beam. The results summarized in Table 14.2.2 apply here if the number n of degrees of freedom there is replaced by $n + q + 2s$.

14.3.2 System Setup and Eigensolutions

The Toolbox of this chapter has a function `setbeam2` for specifying a combined beam, and for computing its eigensolutions; see Window 3.1.

Window 3.1. Function setbeam2

MATLAB Function: **setbeam2**

Purpose:
To set up a combined beam, and to compute its eigensolutions.

Synopsis:

```
setbeam2(rou, EI, L, Dmp_Spec, BC_Spec, SMD_Spec, Osc_Spec,
Rbd_Spec, n_mode)
```

Description:
`setbeam(rou, EI, L, Dmp_Spec, BC_Spec, SMD_Spec, Osc_Spec, Rbd_Spec, n_mode)` inputs linear density `rou`, bending stiffness `EI`, and length `L` of the beam; specifies the damping in the beam by a vector `Dmp_Spec`; sets up the beam boundary conditions by a vector `BC_Spec` according to Table 14.2.3; specifies the attached SMDs by a matrix `SMD_Spec`; specifies the oscillators by a matrix `Osc_Spec`; specifies the rigid bodies by a matrix `Rbd_Spec`; and sets up the number `n_mode` of terms in the modal series (14.40). In addition, `setbeam` computes the eigenvalues and eigenvectors of the beam structure and plots the spatial distributions of the first four mode shapes. By default, `n_mode` = 8. If the beam has no damping, simply replace `Dmp_Spec` by 0 or [ ] as a placeholder. Also, `SMD_Spec`, `Osc_Spec`, and `Rbd_Spec` can be replaced by 0 or [ ] as a placeholder if the beam does not have an SMD set and/or oscillator and/or rigid body. For example, `setbeam2(rou, EI, L, Dmp_Spec, BC_Spec, 0, 0, Rbd_Spec)` sets up a beam combined with rigid bodies only.

Function `setbeam2` in Window 3.1 specifies a combined beam by the following arguments:

(i) The specification of `Dmp_Spec`, `BC_Spec`, and `SMD_Spec` is the same as that for function `setbeam`; see the comments that follow Window 2.1.

(ii) The q oscillators are specified by a q-by-4 matrix

$$\texttt{Osc\_Spec} = \begin{bmatrix} \xi_1 & \hat{k}_1 & \hat{m}_1 & \hat{c}_1 \\ \xi_2 & \hat{k}_2 & \hat{m}_2 & \hat{c}_2 \\ \vdots & \vdots & \vdots & \vdots \\ \xi_q & \hat{k}_q & \hat{m}_q & \hat{c}_q \end{bmatrix}$$

where the jth row of the matrix specifies the jth oscillator located at $x = \xi_j$.

(iii) The s rigid bodies are specified by an s-by-9 matrix

$$\texttt{Rbd\_Spec} = \begin{bmatrix} M_1 & I_1 & b_1 & x_{L1} & k_{L1} & c_{L1} & x_{R1} & k_{L1} & c_{L1} \\ M_2 & I_2 & b_2 & x_{L2} & k_{Ls} & c_{L2} & x_{R2} & k_{L2} & c_{L2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M_s & I_s & b_s & x_{Ls} & k_{Ls} & c_{Ls} & x_{Rs} & k_{Ls} & c_{Ls} \end{bmatrix}$$

where the jth row of the matrix specifies the jth rigid body mounted at points x_{Lj} and x_{Rj}.

Note: After function `setbeam2` is executed, functions `plotmsh` and `animode` given in Windows 2.2 and 2.3 can be called to illustrate and animate modes of vibration of a combined beam structure. To show the information of the combined beam, in the MATLAB command window, type

```
>> systinfo
```

EXAMPLE 3.1

In Fig. 14.3.3, a simply supported undamped beam is connected to a mass-spring subsystem at $x = a$. Let the system parameters be $\rho = 1$, $EI = 1$, $L = 2$, $a = 1$, $k = 60$, $m = 32$.

```
>> rou = 1; EI = 1; L = 2; BC_Spec = [1 1];
>> a = 1; k = 60; m = 32; Osc_Spec = [a k m 0];
>> setbeam2(rou, EI, L, 0, BC_Spec, 0, Osc_Spec)
```

yields the natural frequencies of the combined beam

```
Eigenvalues of the Combined Beam Structure
-----------------------------------------
The system is undamped with natural frequencies:
  mode 1: omega = 0.40782 rad/s
  mode 2: omega = 7.7028 rad/s
  mode 3: omega = 9.8696 rad/s
  mode 4: omega = 23.6433 rad/s
  mode 5: omega = 39.4784 rad/s
```

```
mode 6: omega = 62.1835 rad/s
mode 7: omega = 88.8264 rad/s
mode 8: omega = 121.154 rad/s
mode 9: omega = 157.9137 rad/s
```

and the first four mode shapes plotted in Fig. 14.3.4.

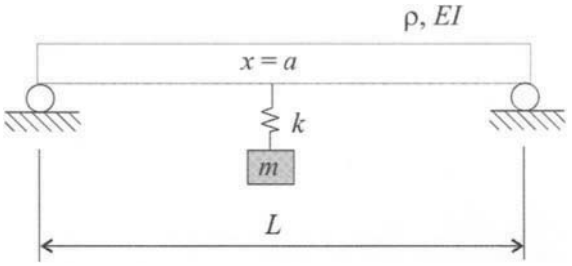


FIGURE 14.3.3

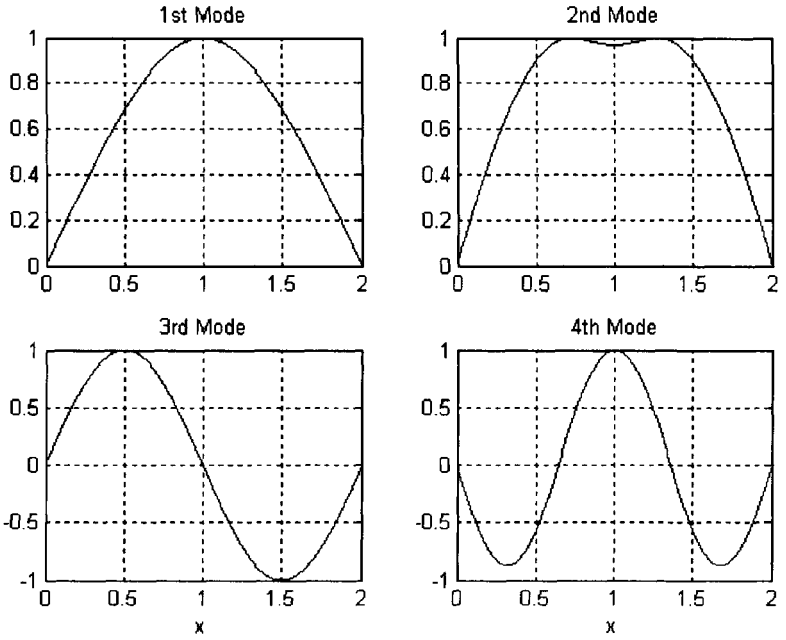


FIGURE 14.3.4

14.3.3 Transient Response

Consider the combined beam in Fig. 14.3.1, which is under external and initial disturbances. The time response of the beam structure is governed by Eqs. (14.34) to (14.39), along with the

boundary conditions listed in Table 14.2.1, and the initial conditions

$$\text{for the beam} \qquad w(x,0) = a_0(x), \quad \frac{\partial}{\partial t} w(x,0) = b_0(x), \quad 0 \le x \le L \tag{14.50a}$$

$$\text{for the oscillators} \qquad \{z_O(0)\} = 0, \quad \{\dot{z}_O(0)\} = 0 \tag{14.50b}$$

$$\text{for rigid bodies} \qquad \{z_R(0)\} = 0, \quad \{\dot{z}_R(0)\} = 0 \tag{14.50c}$$

where $a_0(x)$ and $b_0(x)$ are the initial displacement and velocity of the beam, respectively. Without loss of generality, it has been assumed that the oscillators and rigid bodies are initially at rest. By the modal series (14.40), the above equations are reduced to Eq. (14.41), which is subject to the initial conditions

$$\{q_a(0)\} = \begin{pmatrix} \{U_0\} \\ \{0\} \\ \{0\} \end{pmatrix}, \quad \{\dot{q}_a(0)\} = \begin{pmatrix} \{V_0\} \\ \{0\} \\ \{0\} \end{pmatrix} \tag{14.51}$$

where $\{0\}$ is a zero vector of proper order, and

$$\{U_0\} = \begin{pmatrix} q_1(0) \\ \vdots \\ q_n(0) \end{pmatrix}, \quad \{V_0\} = \begin{pmatrix} \dot{q}_1(0) \\ \vdots \\ \dot{q}_n(0) \end{pmatrix}$$

The $q_k(0)$ and $\dot{q}_k(0)$ are the initial values of the kth generalized coordinate, and they are given by

$$q_k(0) = \rho \int_0^L a_0(x)\,\phi_k(x)dx, \quad \dot{q}_k(0) = \rho \int_0^L b_0(x)\,\phi_k(x)dx \tag{14.52}$$

The $q_k(0)$ and $\dot{q}_k(0)$ can be estimated by numerical integration:

$$q_k(0) \approx \frac{\rho L}{np} \sum_{j=1}^{np} a_0(x_j)\,\phi_k(x_j), \quad \dot{q}_k(0) \approx \frac{\rho L}{np} \sum_{j=1}^{np} b_0(x_j)\,\phi_k(x_j) \tag{14.53}$$

where np equally-spaced points $x_j = j \cdot L/np$ along the beam length are used.

Let the external force have the form given in Eq. (14.35). Vector $\{B_f\}$ in Eq. (14.43) is

$$\{B_f\} = \begin{pmatrix} b_1 & b_2 & \cdots & b_n \end{pmatrix}^T \tag{14.54}$$

where $b_k = \int_0^L b(x)\phi_k(x)dx$. Three types of force distribution are considered here:

(a) Pointwise force at x_f (Fig. 14.3.5a): $F(x,t) = \delta(x - x_f)f(t)$, where $\delta(\cdot)$ is the delta function. In this case

$$b_k = \phi_k(x_f) \tag{14.55a}$$

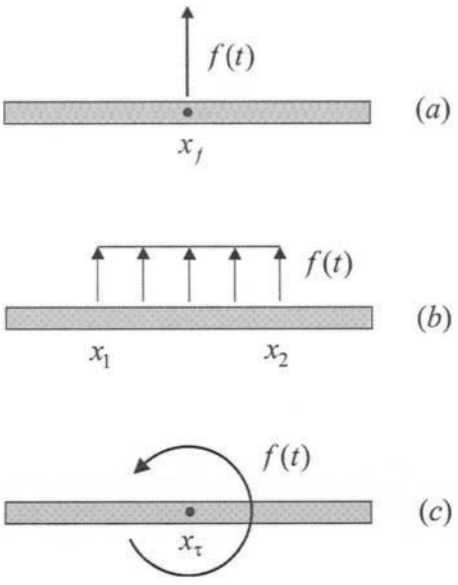


FIGURE 14.3.5 Three types of spatial distribution of external forces.

(b) Uniformly distributed force in the region $x_1 \leq x \leq x_2$ (Fig. 14.3.5b): $F(x,t) = \{h(x-x_1) - h(x-x_2)\}f(t)$, where $h(\cdot)$ is the unit step function. In this case,

$$b_k = \int_{x_1}^{x_2} \phi_k(x)dx \tag{14.55b}$$

(c) Pointwise torque at $x = x_\tau$ (Fig. 14.3.5c): $F(x,t) = -\frac{d\delta(x-x_\tau)}{dx}f(t)$. In this case,

$$b_k = -\int_0^L \frac{d\delta(x-x_\tau)}{dx}u_k(x)dx = \frac{du_k(x_\tau)}{dx} \tag{14.55c}$$

According to the above discussion, transient response analysis of a combined beam is boiled down to the solution of Eq. (14.41) with the initial conditions (14.51). Once the generalized coordinates q_k are determined, the response of the beam at any point can be computed by the modal series (14.40).

The transient response of a combined beam can be computed via numerical integration. To this end, Eq. (14.41) is cast to the first-order state equation

$$\{\dot{\eta}(t)\} = \Gamma\,(t,\ \{\eta(t)\}), \quad \{\dot{\eta}(0)\} = \{\eta_0\} \tag{14.56}$$

where

$$\{\eta(t)\} = \begin{Bmatrix}\{q_a(t)\}\\ \{\dot{q}_a(t)\}\end{Bmatrix}, \quad \{\eta_0\} = \begin{Bmatrix}\{q_a(0)\}\\ \{\dot{q}_a(0)\}\end{Bmatrix}$$
$$\Gamma\,(t,\{\eta(t)\}) = [A]\{\eta(t)\} + \{B\}f(t) \tag{14.57}$$

with

$$[A] = \begin{bmatrix} [0] & [I] \\ -[M]^{-1}[K] & -[M]^{-1}[D] \end{bmatrix}, \quad \{B\} = \begin{pmatrix} \{0\} \\ [M]^{-1}\{B_a\} \end{pmatrix} \tag{14.58}$$

The state equation (14.56) is solved by the fixed-step Runge-Kutta method of order four:

$$\{\eta_{k+1}\} = \{\eta_k\} + \frac{h}{6}(\{f_1\} + 2\{f_2\} + 2\{f_3\} + \{f_4\}) \tag{14.59}$$

where $\{\eta_k\} = \{\eta(t_k)\}$ with $t_k = kh$, $k = 0, 1, 2, \ldots$, and h being the step size, and

$$\begin{aligned} \{f_1\} &= \Gamma(t_k, \{z_k\}) \\ \{f_2\} &= \Gamma\left(t_k + \frac{h}{2}, \{z_k\} + \frac{h}{2}\{f_1\}\right) \\ \{f_3\} &= \Gamma\left(t_k + \frac{h}{2}, \{z_k\} + \frac{h}{2}\{f_2\}\right) \\ \{f_4\} &= \Gamma(t_k + h, \{z_k\} + h\{f_3\}) \end{aligned} \tag{14.60}$$

The Toolbox of this chapter provides functions `trbeam2` and `trbeam2s` for transient response analysis of combined beams (see Windows 3.2 and 3.3) and function `beammovie2` for animation of vibration of combined beams (see Window 3.4).

Window 3.2. Function `trbeam2`

MATLAB Function: **`trbeam2`**

Purpose:
To plot the transient response of a combined beam at a point by the Runge-Kutta numerical algorithm (14.59).

Synopsis:

```
trbeam2(xr, Load_Spec, IC)
trbeam2(xr, Load_Spec, IC, tf, npts)
[y, t, qa]= trbeam2(xr, Load_Spec, IC, tf, npts)
```

Description:
`trbeam2(xr, Load_Spec, IC)` plots the time history of the transient displacement of a combined beam at location `xr`. The beam is subject to an external load specified by a vector `Load_Spec` and initial disturbances specified by a two-row matrix `IC`. The formation of `Load_Spec` and `IC` is discussed after this window. `Load_Spec` or `IC` can be replaced by 0 or [ ] as a placeholder if the external load or initial disturbances do not exist. In other words, the forced response of the beam (with zero initial disturbances) is obtained by `trbeam2(xr, Load_Spec)`; the free response of the beam (with zero external force) is determined by `trbeam2(xr, 0, IC)` or `trbeam2(xr, [], IC)`.

Window 3.2. Function trbeam2 (*Continued*)

`trbeam2(xr, Load_Spec, IC, tf, npts)` computes the transient displacement of the combined beam in the time region $0 \leq t \leq$ `tf`, where `npts` is the number of equally spaced points in the time region for simulation. By default, `tf` = $\frac{\pi}{4\omega_H}$(`npts` − 1) and `npts` = 2001, where ω_H is the highest natural or damped frequency of the combined beam.

`[y, t, qa] = trbeam2(xr, Load_Spec, IC, tf, npts)` returns the beam transient displacement in a vector `y`, the times of computation in vector `t`, and the augmented displacement vector $\{q_a(t)\}$ of Eq. (14.42) in a matrix `qa`. The beam displacement can be plotted by the MATLAB function `plot(t,y)`.

Note 1. The usage of function `trbeam2` requires specification of `Load_Spec` and `IC`. The specification of `Load_Spec` is given in Table 14.3.1 for nine types of external loads, where $u(t - T_d)$ is the delayed unit step function, which is one for $t \geq T_d$ and zero otherwise, and T_d is a nonnegative time delay. According to the table, `Load_Spec` is of the form

```
Load_Spec = [load_Id par1 par2 ...]
```
(14.61)

where `load_Id` is a load identification number, and `par1, par2 ...` are parameters describing the location and pattern of the external load. The `IC` is a 2-by-np matrix specifying the initial disturbances of the beam at np points:

$$IC = \begin{bmatrix} a_0(x_1) & a_0(x_2) & \cdots & a_0(x_{np}) \\ b_0(x_1) & b_0(x_2) & \cdots & b_0(x_{np}) \end{bmatrix} \tag{14.62}$$

where $a_0(x_j)$ and $b_0(x_j)$ are the initial displacement and velocity of the beam at point x_j.

Note 2. The step size h of the Runge-Kutta algorithm, as shown in Eq. (14.59), is given by

$$h = \frac{\texttt{tf}}{\texttt{npts} - 1} \tag{14.63}$$

which must be small enough to avoid divergence in numerical integration. As a rule of thumb,

$$h < \frac{1}{2}\frac{2\pi}{\omega_H} = \frac{\pi}{\omega_H} \tag{14.64}$$

where ω_H is the highest natural or damped frequency of the beam modeled by Eq. (14.41). To reduce the step size, either increase the number `npts` of time points for simulation or decrease the computation time `tf`.

Note 3. Assume that the combined beam in consideration carries q oscillators and s rigid bodies, as shown in Fig. 14.3.1. The total number of degrees of freedom of the beam modeled by Eq. (14.41) is $N_{\text{total}} = n + q + 2s$, where n is the number of terms in Eq. (14.40). In this case, the output argument `qa` of function `trbeam2` is an N_{total}-by-`npts` matrix. By Eq. (14.42), the first n rows of the matrix contain the time history of the n generalized

TABLE 14.3.1 External Loads and Load Specification Vector Load_Spec

External Load	*F(x,t)*	Load_Spec
(L0) Pointwise impulse load	$I_0\delta(t)\delta(x-x_f)$	$[0\ x_f\ I_0]$
	$I_0\delta(t-T_d)\delta(x-x_f)$	$[0\ x_f\ I_0\ T_d]$
(L1) Pointwise step load	$F_0\delta(x-x_f)$	$[1\ x_f\ F_0]$
	$F_0u(t-T_d)\delta(x-x_f)$	$[1\ x_f\ F_0\ T_d]$
(L2) Pointwise sinusoidal load	$A\sin(\omega t+\phi_0)\delta(x-x_f)$	$[2\ x_f\ A\ \omega\ \phi_0]$ ω in rad/s, ϕ_0 in degrees
	$A\sin(\omega t+\phi_0)u(t-T_d)\delta(x-x_f)$	$[2\ x_f\ A\ \omega\ \phi_0\ T_d]$ ω in rad/s, ϕ_0 in degrees
(L10) Uniformly distributed impulse load	$I_0\delta(t)$, for $x_1\le x\le x_1$	$[10\ x_1\ x_2\ I_0]$
	$I_0\delta(t-T_d)$, for $x_1\le x\le x_1$	$[10\ x_1\ x_2\ I_0\ T_d]$
(L11) Uniformly distributed step load	F_0, for $x_1\le x\le x_1$	$[11\ x_1\ x_2\ F_0]$
	$F_0u(t-T_d)$, for $x_1\le x\le x_1$	$[11\ x_1\ x_2\ F_0\ T_d]$
(L12) Uniformly distributed sinusoidal load	$A\sin(\omega t+\phi_0)$, for $x_1\le x\le x_1$	$[12\ x_1\ x_2\ A\ \omega\ \phi_0]$ ω in rad/s, ϕ_0 in degrees
	$A\sin(\omega t+\phi_0)u(t-T_d)$, for $x_1\le x\le x_1$	$[12\ x_1\ x_2\ A\ \omega\ \phi_0\ T_d]$ ω in rad/s, ϕ_0 in degrees
(L20) Pointwise impulse torque	$-I_0\delta(t)\dfrac{d\delta(x-x_\tau)}{dx}$	$[20\ x_\tau\ I_0]$
	$-I_0\delta(t-T_d)\dfrac{d\delta(x-x_\tau)}{dx}$	$[20\ x_\tau\ I_0\ T_d]$
(L21) Pointwise step torque	$-\tau_0\dfrac{d\delta(x-x_\tau)}{dx}$	$[21\ x_\tau\ \tau_0]$
	$-\tau_0\dfrac{d}{dx}\delta(x-x_\tau)u(t-T_d)$	$[21\ x_\tau\ \tau_0\ T_d]$
(L22) Pointwise sinusoidal torque	$-A\sin(\omega t+\phi_0)\dfrac{d\delta(x-x_\tau)}{dx}$	$[22\ x_\tau\ A\ \omega\ \phi_0]$ ω in rad/s, ϕ_0 in degrees
	$-A\sin(\omega t+\phi_0)\dfrac{d}{dx}\delta(x-x_\tau)u(t-T_d)$	$[22\ x_\tau\ A\ \omega\ \phi_0\ T_d]$ ω in rad/s, ϕ_0 in degrees

coordinates beam $q_1(t), q_2(t), \cdots, q_n(t)$; the next q rows contain the oscillator displacements $y_1(t), y_2(t), \cdots, y_q(t)$; and the last $2s$ rows contain the translations and rotations of the rigid bodies $z_1(t), \theta_1(t), \cdots, z_s(t), \theta_s(t)$. With `qa` and `t` as the output data of `trbeam2`, the vibration of the oscillators and rigid bodies can be plotted by the MATLAB function `plot`. For instance, to illustrate the displacement $y_j(t)$ of the jth oscillator, execute the command `plot(t, y(n+j, :))` and to display the rotation $\theta_k(t)$ of the kth rigid body, execute the command `plot(t, y(n+q+2*k, :))`.

EXAMPLE 3.2

Consider the same combined beam as given in Example 3.1. Assume that the beam has viscous damping, $d_v = 0.7$. The beam is subject to a pointwise step load $F_0 = -1$ at $x_f = 0.6$. The forced vibration of the beam at $x = 1.2$ is plotted in Fig. 14.3.6 by the following commands:

```
>> setbeam2(1, 1, 2, 0.7, [1 1], 0, [1 60 32 0])
>> Load_Spec = [1 0.6 -1];
>> trbeam2(1.2, Load_Spec, 0, 20, 5001)
```

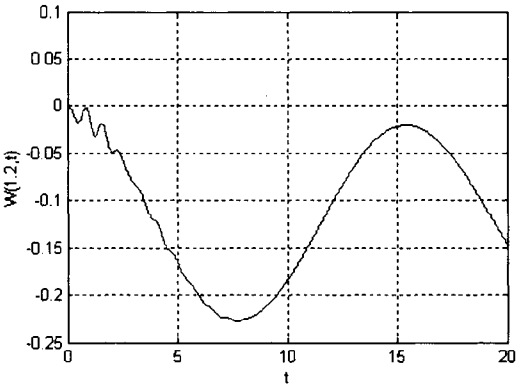


FIGURE 14.3.6

Window 3.3. Function trbeam2s

MATLAB Function: **trbeam2s**

Purpose:
To plot the spatial distribution of the displacement of a combined beam at a specific time by the Runge-Kutta numerical algorithm (14.59).

Synopsis:

```
trbeam2s(t1, Load_Spec, IC)
trbeam2s(t1, Load_Spec, IC, npts, n_xpts)
[y, x] = trbeam2s(t1, Load_Spec, IC, npts, n_xpts)
```

Description:
`trbeam2s(t1, Load_Spec, IC)` plots the spatial distribution of the displacement of a combined beam at a specified time `t1`. The beam is subject to an external load specified by a vector `Load_Spec` and initial disturbances specified by a two-row matrix `IC`. The formation of `Load_Spec` and `IC` follows that for function `trbeam2`. `Load_Spec` or `IC` is replaced by 0 or [ ] if the external load or initial disturbances do not exist.

Window 3.3. Function trbeam2s (*Continued*)

`trbeam2s(t1, Load_Spec, IC, npts, n_xpts)` plots the beam displacement distribution in a specified manner, where `npts` is the number of equally spaced time points in the region `[0, t1]` for numerical integration, and `n_xpts` is the number of equally-spaced points along the beam length at which the beam response is computed. By default, `npts` is an integer that is rounded up from $4\omega_H \mathtt{tf}/\pi + 2$ with ω_H being the highest natural or damped frequency of the combined beam, and `n_xpts` = 501.

`[y, x] = trbeam2s(t1, Load_Spec, IC, npts, n_xpts)` returns the beam displacement in a vector `y`, the spatial points of computation in a vector `x`. The spatial distribution of the beam can be plotted by the MATLAB function `plot(x,y)`.

Window 3.4. Function beammovie2

MATLAB Function: **beammovie2**

Purpose:
To play animated vibration of a combined beam that is computed by the Runge-Kutta algorithm (14.59).

Synopsis:

```
beammovie2(Load_Spec, IC)
beammovie2(Load_Spec, IC, tf, n_frames, n_control)
F = beammovie2(Load_Spec, IC, tf, n_frames, n_control)
```

Description:
`beammovie2(Load_Spec, IC)` plays animated vibration of a combined beam subject to an external load specified by a vector `Load_Spec` and initial disturbances specified by a two-row matrix `IC`. The formation of `Load_Spec` and `IC` follows that for function `trbeam2`. `Load_Spec` or `IC` can be replaced by 0 or `[]` if the external load or initial disturbances do not exist.

`beammovie2(Load_Spec, IC, tf, n_frames, n_control)` plays animated beam vibration in a special manner, where `tf` is total animation time, `n_frames` is the total number of frames, and `n_control` is an integer controlling the step size h of the numerical integration by

$$h = \frac{\mathtt{tf}}{(\mathtt{n\_frames} - 1) \times \mathtt{n\_control}}$$

By default, $\mathtt{tf} = \frac{\pi}{6\omega_H} \times (\mathtt{n\_frames} - 1) \times \mathtt{n\_control}$ with ω_H being the highest natural or damped frequency of the combined beam, `n_frames` = 101, and `n_control` = 100.

`F = beammovie2(Load_Spec, IC, tf, n_frames, n_control)` returns a set of frames of the animated beam vibration in a matrix `F`, which can be played by the MATLAB function `movie(F)`.

Functions `trbeam2`, `trbeam2s`, and `beammovie2` can all be used for transient analysis of constrained beams as discussed in Section 14.2, which are a special case of combined beams.

EXAMPLE 3.3

In Fig. 14.3.7, a rigid body is mounted on a clamped-pinned beam. The parameters of the combined beam are

$$\rho = 2.5, \quad EI = 120, \quad L = 4$$
$$M = 0.1, \quad I = 0.05, \quad b = 0.05$$
$$x_1 = 1.7, \quad k_1 = 20, \quad c_1 = 0.9$$
$$x_2 = 1.8, \quad k_2 = 18, \quad c_2 = 1.2$$

The combined beam is set up by

```
>> rbd = [0.1 0.05 0.05 1.7 20 0.9 1.8 18 1.2];
>> setbeam2(2.5, 120, 4, 0, [2 1], 0, 0, rbd)
```

which yields the eigenvalues of the beam as follows

```
Eigenvalues of the Combined Beam
--------------------------------
  The system is nonproportionally damped with eigenvalues:
  mode 1:  lambda = -0.053414+1.3754i, -0.053414-1.3754i
  mode 2:  lambda = -0.0024839+6.6136i, -0.0024839-6.6136i
  mode 3:  lambda = -10.6578+16.6627i, -10.6578-16.6627i
  mode 4:  lambda = -0.124852+21.5331i, -0.124852-21.5331i
  mode 5:  lambda = -0.0653259+45.1302i, -0.0653259-45.1302i
  mode 6:  lambda = -0.18711+77.1825i, -0.18711-77.1825i
  mode 7:  lambda = -0.01477384+117.7936i, -0.01477384-117.7936i
  mode 8:  lambda = -0.2000873+166.9348i, -0.2000873-166.9348
```

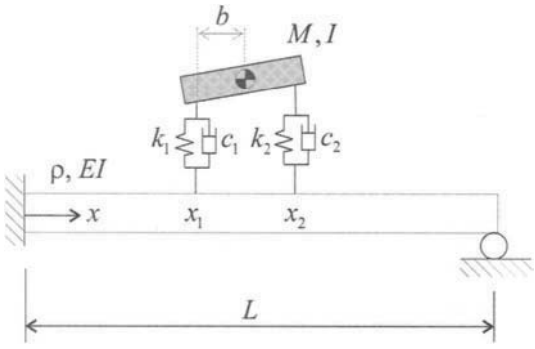


FIGURE 14.3.7

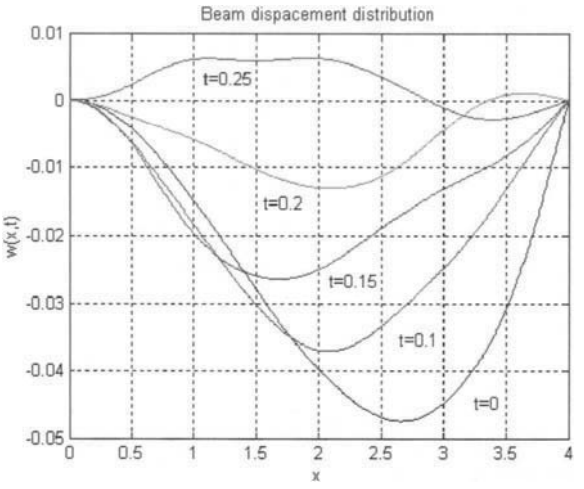


FIGURE 14.3.8

Now assume that the combined beam has the initial displacement $w(x,0) = -0.005x^2(L-x)$, $0 \le x \le L$, and zero initial velocity. The spatial distribution of the free response of the beam at $t = 0$, 0.1, 0.15, 0.2, and 0.25 are plotted in Fig. 14.3.8 by

```
>> xx = linspace(0, 4, 501);
>> x0 = 0.005*xx.^2.*(xx -4); v0 = zeros(1,501);
>> IC = [x0; v0];
>> [y0, x] = trbeam2s(0, 0, IC);
>> y1 = trbeam2s(0.1, 0, IC);
>> y2 = trbeam2s(0.15, 0, IC);
>> y3 = trbeam2s(0.2, 0, IC);
>> y4 = trbeam2s(0.25, 0, IC);
>> clf
>> plot(x, y0, x, y1, x, y2, x, y3, x, y4)
>> grid; title('Beam displacement distribution')
>> xlabel('x'); ylabel('w(x,t)')
```

Also, the free response of the combined beam can be animated by `beammovie2(0, IC)`.

14.3.4 Frequency Response

A combined beam is subject to a pointwise sinusoidal force $f_0\, e^{j\omega t}\, \delta(x - x_f)$, where f_0 and x_f are the amplitude and location of the force, ω is the excitation frequency, and $j = \sqrt{-1}$. Under such a load, Eq. (14.41) becomes

$$[M]\{\ddot{q}_a(t)\} + [D]\{\dot{q}_a(t)\} + [K]\{q_a(t)\} = \{B_a\} f_0\, e^{j\omega t} \tag{14.65}$$

The steady-state solution of the above equation is

$$\{q_a(t)\} = \{Q\}\, e^{j\omega t} \tag{14.66}$$

with

$$\{Q\} = f_0 \left(-\omega^2 [M] + j\omega [D] + [K]\right)^{-1} \{B_a\} \tag{14.67}$$

Partition $\{Q\}$ as

$$\{Q\} = \begin{Bmatrix} \{Q_b\} \\ \{Q_O\} \\ \{Q_R\} \end{Bmatrix} \tag{14.68}$$

where the n-vector $\{Q_b\}$ contains the n modal coordinates of the beam; the q-vector $\{Q_O\}$ represents the response of the q oscillators; and the $2s$-vector $\{Q_R\}$ represents the response of the s rigid bodies in both translation and rotation. It follows that the steady-state displacement of the beam is

$$w(x,t) = W(x,\omega)e^{j\omega t} \tag{14.69}$$

where

$$W(x,\omega) = [\Phi(x)]^T \{Q_b\} \tag{14.70}$$

The steady-state displacements of the oscillators are given by

$$\{z_O(t)\} = \{Q_O\} e^{j\omega t} \tag{14.71}$$

while the steady-state displacements of the rigid bodies are given by

$$\{z_R(t)\} = \{Q_R\} e^{j\omega t} \tag{14.72}$$

The $W(x,\omega)$, $\{Q_O\}$, and $\{Q_R\}$ are called the *frequency response* of the beam, oscillators, and rigid bodies, respectively.

The Toolbox of this chapter provides a function `frfbeam2` for plotting the frequency response of a combined beam subject to a pointwise sinusoidal load; see Window 3.5.

Window 3.5. Function `frfbeam2`

MATLAB Function: **`frfbeam2`**

Purpose:
To plot the magnitude and phase of steady-state displacement of a combined beam subject to a pointwise sinusoidal load.

Synopsis:

```
frfbeam2(Ix, xf, f0)
frfbeam2(Ix, xf)
frfbeam2(Ix, xf, f0, omg0, omgf, npts)
[y, freq] = frfbeam2(Ix, xf, f0, omg0, omgf, npts)
```

Window 3.5. Function `frfbeam2` (*Continued*)

Description:

`frfbeam2(Ix, xf, f0)` plots the magnitude and phase of the steady-state displacement of the combined beam under a sinusoidal load that has an amplitude `f0` and is applied at `xf`. Here `Ix` is a vector specifying the location of frequency response with the options:

`Ix = xr` for steady-state displacement of the beam at location `xr`;
`Ix = [1 k]` for steady-state displacement of the kth oscillator; and
`Ix = [2 k]` for steady-state displacements of the kth rigid body.

`frfbeam2(Ix, xf)` plots the frequency response for `f0 = 1`.

`frfbeam2(Ix, xf, f0, omg0, omgf, npts)` plots the frequency response in a frequency region `[omg0, omgf]`, where `npts` is the number of frequency points in the region for simulation. By default, `omg0 = 0`, `omgf` is a number larger than the third natural or damped frequency of the beam without SMDs, oscillators, and rigid bodies, and `npts = 1001`.

`[y, freq ] = frfbeam2(Ix, xf, f0, omg0, omgf, npts)` returns the computed frequency response in a matrix `y` and frequencies of simulation in a vector `freq`. With `y` and `freq`, the frequency response can be plotted by the MATLAB function `plot` as follows:

For the response of the beam at `xr`:
`plot(freq, y(1,:))` for amplitude, `plot(freq, y(2,:))` for phase
For the response of the kth oscillator:
`plot(freq, y(1,:))` for amplitude, `plot(freq, y(2,:))` for phase
For the translational response of the kth rigid body:
`plot(freq, y(1,:))` for amplitude, `plot(freq, y(2,:))` for phase
For the rotational response of the kth rigid body:
`plot(freq, y(3,:))` for amplitude, `plot(freq, y(4,:))` for phase

EXAMPLE 3.4

In Fig. 14.3.9, a rigid body is mounted on a clamped-clamped beam. The parameters of the combined beam are given as follows

$$\begin{aligned} \rho &= 1, & EI &= 1, & L &= 2 \\ M &= 0.4, & I &= 0.1, & b &= 0.05 \\ x_1 &= 0.9, & k_1 &= 10, & c_1 &= 0.3 \\ x_2 &= 1.1, & k_2 &= 10, & c_2 &= 0.3 \end{aligned}$$

The beam has viscous damping with $d_v = 0.01$. A sinusoidal force of unit amplitude ($f_0 = 1$) is applied to the beam at location $x_f = 0.5$.

```
>> rou = 1; EI = 1; L = 2; dv = 0.01; BC_Spec = [2 2];
>> Rbd_Spec = [0.4 0.1 0.05 0.9 10 0.3 1.1 10 0.3];
```

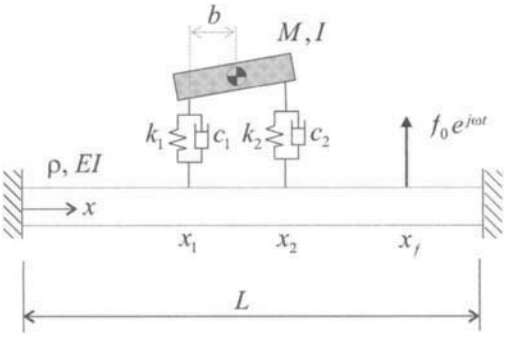


FIGURE 14.3.9

```
>> setbeam2(rou, EI, L, dv, BC_Spec, 0, 0, Rbd_Spec)
>> xr = 1.1; xf = 0.5; f0 = 1;
>> frfbeam2(xr, xf, f0, 0, 80)
```

plots the frequency response of the beam at location $x_r = 1.1$ in Fig. 14.3.10.

For comparison, the frequency response of the beam without the rigid body is plotted by

```
>> setbeam2(rou, EI, L, dv, BC_Spec)
>> frfbeam2(xr, xf, f0, 0, 80)
```

which yield Fig. 14.3.11. Furthermore, by

```
>> setbeam2(rou, EI, L, dv, BC_Spec, 0, 0, Rbd_Spec)
>> frfbeam2([2 1], 0.5,1,0, 80, 4001)
```

the frequency response of the rigid body in both translation and rotation is plotted in Figs. 14.3.12 (a) and (b), respectively.

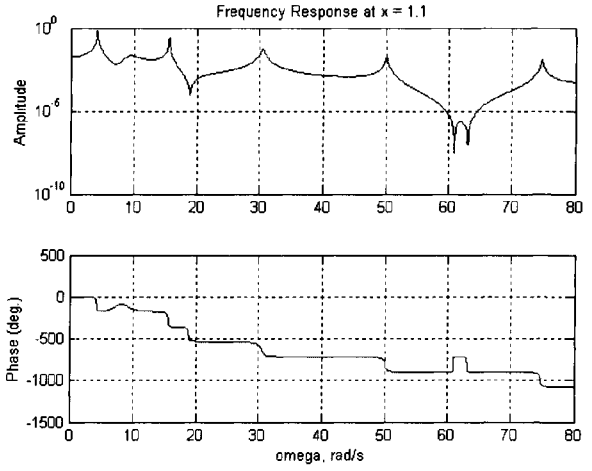


FIGURE 14.3.10 Frequency response of the beam carrying a rigid body.

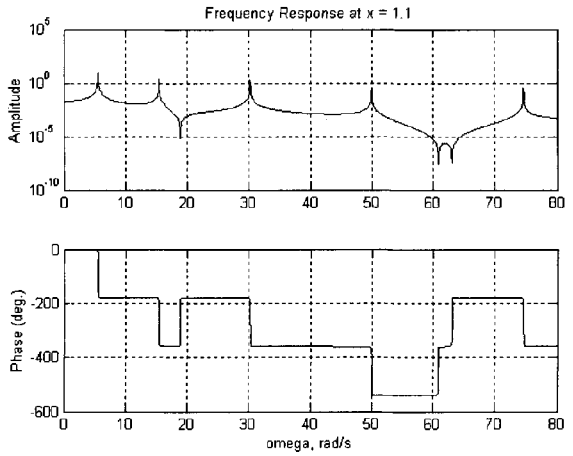


FIGURE 14.3.11 Frequency response of the beam without a rigid body.

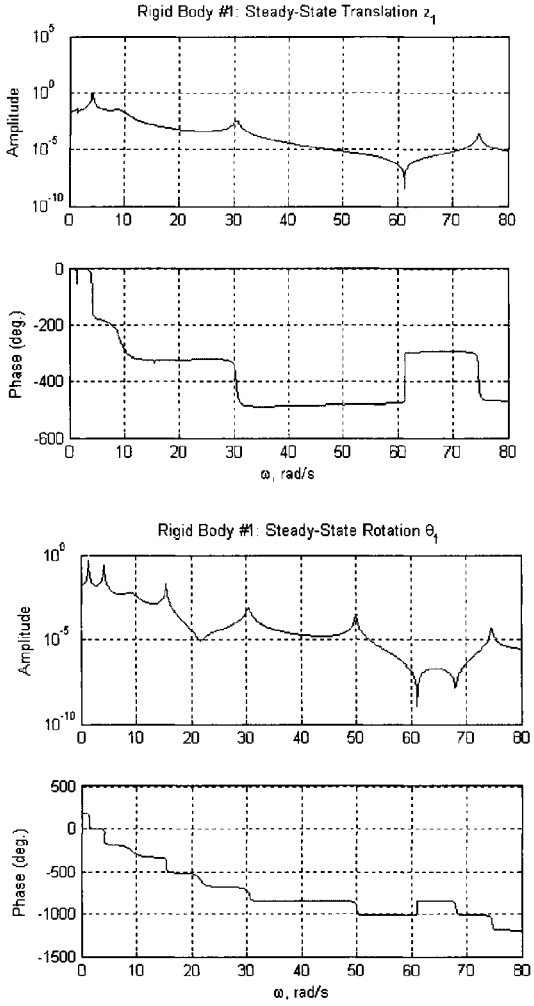


FIGURE 14.3.12 Steady-state response of the rigid body: (a) translation; (b) rotation.

14.3.5 Oscillators for Vibration Absorption

An oscillator that is connected to a beam can serve as a dynamic vibration absorber. In vibration absorption, an added oscillator is designed such that possible resonant vibration of the host structure (beam) is avoided. As a rule of thumb, the spring and mass parameters (k, m) of an absorber (oscillator) are tuned such that the natural frequency of the oscillator ($\sqrt{k/m}$) nearly coincides with a resonance frequency or natural frequency of the beam. Without damping, the vibration absorber so designed is effective only in a narrow frequency band around a resonance frequency of the beam. Accordingly, damping is often introduced to the absorber to improve its bandwidth of operation in vibration reduction. Refer to Chapter 11 for the basic concept of dynamic vibration absorption.

The utility of oscillators as vibration absorbers for reduction of beam harmonic response is illustrated through the following example.

EXAMPLE 3.5

Consider a cantilever beam ($\rho = 1$, $EI = 1$, $L = 2$) under a harmonic excitation at its tip. The command

```
>> setbeam2(1, 1, 2, 0, [2 3])
```

yields the first four natural frequencies of the beam as follows

$$\omega_1 = 0.879 \text{ rad/s}, \quad \omega_2 = 5.5086 \text{ rad/s}, \quad \omega_3 = 15.4243 \text{ rad/s}, \quad \omega_4 = 30.2255 \text{ rad/s}$$

By

```
>> frfbeam2(0.8, 2, 1, 0, 40)
```

the frequency response of the beam at $x = 0.8$ is plotted in Fig. 14.3.13.

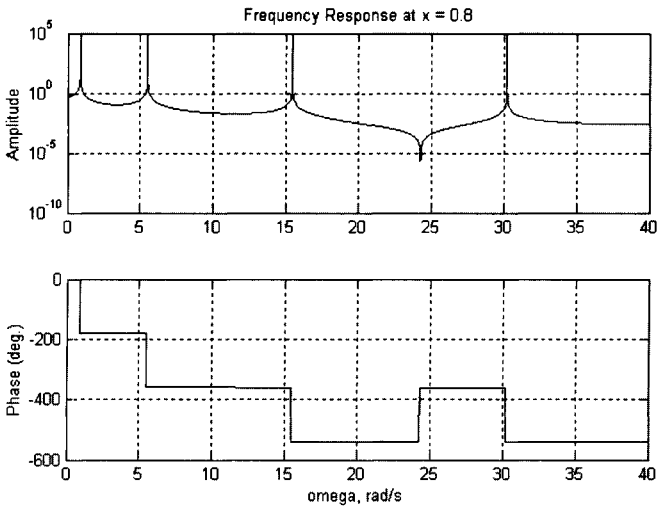


FIGURE 14.3.13 Frequency response of a cantilever beam.

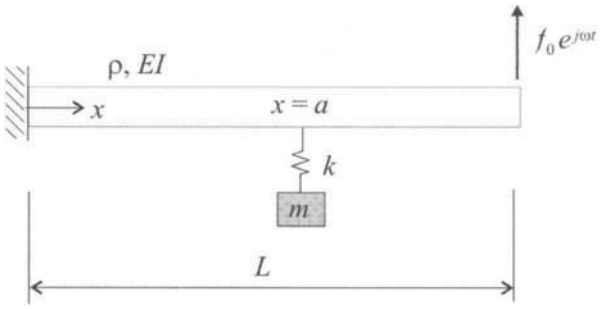


FIGURE 14.3.14

Now, an oscillator is added to the beam at $x = a$ for dynamic vibration absorption, as shown in Fig. 14.3.14. To avoid the resonance at the first natural frequency ω_1, the parameters of the oscillator are selected as

$$a = 1.2, \quad k = 0.16, \quad m = 0.25, \quad c = 0.1$$

The natural frequency of the oscillator is close to ω_1 as $\sqrt{k/m} = 0.8$. The frequency response of the combined beam at $x = 0.8$ is plotted in Fig.14.3.15 by

```
>> setbeam2(1, 1, 2, 0, [2 3], 0, [1.2 0.16 0.25 0.1])
>> frfbeam2(0.8,2, 1, 0, 40)
```

The figure indicates that the attachment of the oscillator to the beam removes the resonance around ω_1.

To avoid resonance at the second natural frequency of the beam, the parameters of the oscillator can be chosen as $a = 0.4$, $k = 7.0225$, $m = 0.25$, $c = 0.45$ such that the natural frequency of the oscillator is close to ω_2.

```
>> setbeam2(1, 1, 2, 0, [2 3], 0, [0.4 7.0225 0.25 0.45])
>> frfbeam2(0.8, 2, 1, 0, 40)
```

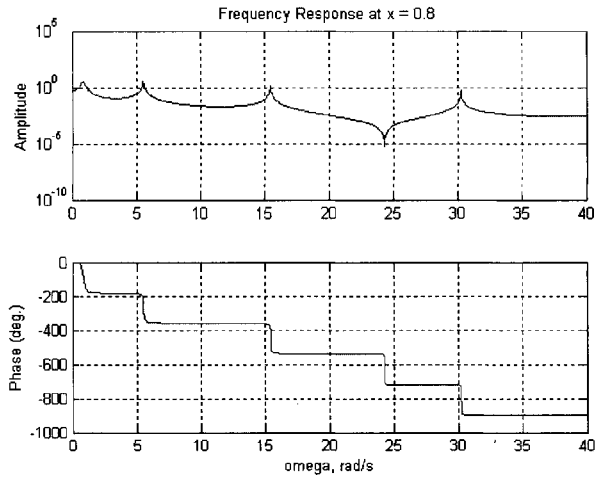


FIGURE 14.3.15 Frequency response of the beam combined with one oscillator, $\sqrt{k/m} = 0.8$.

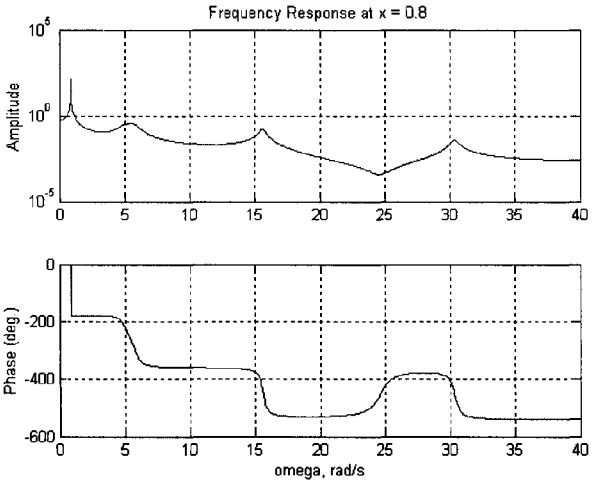


FIGURE 14.3.16 Frequency response of the beam combined with one oscillator, $\sqrt{k/m} = 5.3$.

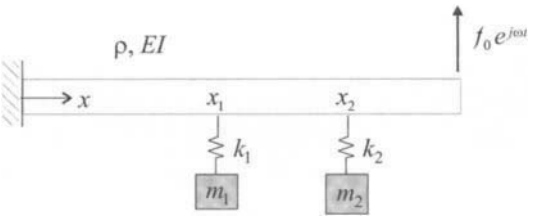


FIGURE 14.3.17 A cantilever beam with two oscillators.

gives the frequency response of the combined beam at $x = 0.8$ in Fig. 14.3.16. As seen from the figure, the resonant peak of the combined beam around ω_2 is removed, but the resonant peak around ω_1 stays.

To remove the first two resonance peaks, consider the beam combined with two oscillators (Fig. 14.3.17), with the parameters

$$x_1 = 1.2, \quad k_1 = 0.16, \quad m_1 = 0.25, \quad c_1 = 0.1$$

$$x_2 = 0.4, \quad k_2 = 7.0225, \quad m_2 = 0.25, \quad c_2 = 0.45$$

Note that these oscillators have been used individually for vibration reduction around ω_1 and ω_2. These oscillators are now added to the beam simultaneously. The commands

```
>> Osc_Spec = [1.2 0.16 0.25 0.1; 0.4 7.0225 0.25 0.45];
>> setbeam2(1, 1, 2, 0, [2 3], 0, Osc_Spec)
>> frfbeam2(0.8, 2, 1, 0, 40)
```

plot the frequency response of the combined beam at $x = 0.8$ in Fig. 14.3.18, where the first two resonant peaks are effectively suppressed.

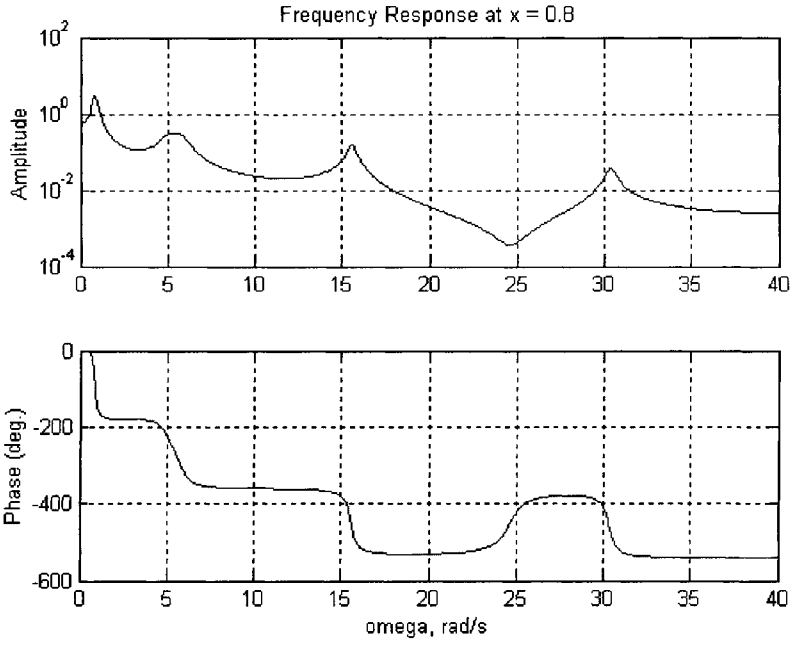


FIGURE 14.3.18

14.4 Stepped Beams

14.4.1 Free Vibration Analysis

This section is concerned with free vibration of stepped beams. A typical stepped beam is an assembly of N uniform Euler-Bernoulli beam segments that are serially connected at N-1 points called *nodes*; see Fig. 14.4.1. The left and right boundaries of the structure are labeled node 0 and node N, respectively. At those nodes, there may be hinges (see node 3 for example), sliding constraints (node 1), springs (node N-1), and lumped masses (node 2). In addition, some beam segments may be constrained by elastic foundation (like the one between nodes 1 and 2).

Beam Elements Denote the kth beam segment by S_k, which is bounded by nodes k-1 and k. The transverse displacement $w_k(x,t)$ of S_k is governed by

$$\rho_k \frac{\partial^2}{\partial t^2} w_k(x,t) + EI_k \frac{\partial^4}{\partial x^4} w_k(x,t) + \mu_k w_k(x,t) = 0, \quad 0 \le x \le L_k \tag{14.73}$$

where ρ_k, EI_k, and L_k are the linear density, bending stiffness, and length of the segment, μ_k is the coefficient of the elastic foundation in S_k, and x is a local spatial coordinate measured from the left end of S_k. In free vibration analysis, the displacement $w_k(x,t)$ in Eq. (14.73) is assumed as

$$w_k(x,t) = u_k(x)e^{j\omega t}, \quad j = \sqrt{-1} \tag{14.74}$$

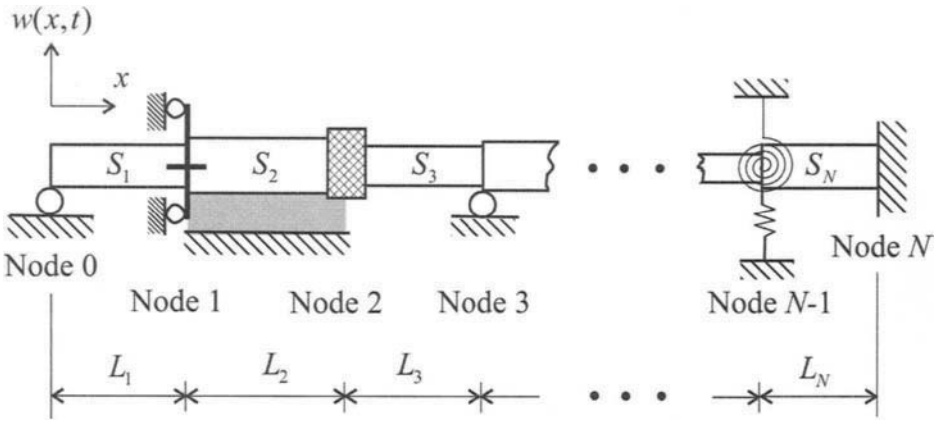


FIGURE 14.4.1 Schematic of a stepped beam.

where ω is a frequency parameter, and $u_k(x)$ is a mode shape of the segment. Substitute Eq. (14.74) into Eq. (14.73) to obtain

$$\frac{d^4}{dx^4}u_k(x) = \beta_k^4 u_k(x), \quad 0 \le x \le L_k \tag{14.75}$$

where

$$\beta_k^4 = \frac{\rho_k \omega^2 - \mu_k}{EI_k} \tag{14.76}$$

Constraints and Segment Interconnection Assume that a constraint is imposed at node k where segments S_k and S_{k+1} are interconnected. Table 14.4.1 lists six types of constraints considered in this chapter, in which k_r is the coefficient of a torsional spring, k_t is the coefficient of a translational spring, m and I are the inertia and moment of inertia of a lumped mass, and I_D is the moment of inertia of a hinged rigid disk. The matching conditions for the displacements and internal forces of S_k and S_{k+1} are given as follows.

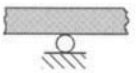
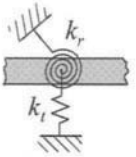
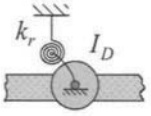
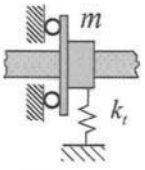
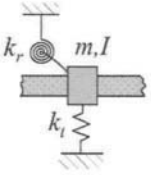
(C1) Hinge:

$$\begin{gathered} u_k(L_k) = 0, \quad u_{k+1}(0) = 0 \\ \frac{d}{dx}u_k(L_k) = \frac{d}{dx}u_{k+1}(0), \quad EI_k\frac{d^2}{dx^2}u_k(L_k) = EI_{k+1}\frac{d^2}{dx^2}u_{k+1}(0) \end{gathered} \tag{14.77a}$$

(C2) Sliding constraint:

$$\begin{gathered} u_k(L_k) = u_{k+1}(0), \quad \frac{d}{dx}u_k(L_k) = 0, \quad \frac{d}{dx}u_{k+1}(0) = 0 \\ EI_k\frac{d^3}{dx^3}u_k(L_k) = EI_{k+1}\frac{d^3}{dx^3}u_{k+1}(0) \end{gathered} \tag{14.77b}$$

TABLE 14.4.1 Specification of Constraints for Stepped Beams

Constraint Type		Constr_j
(C1) Hinge		[node_no 1 0 0 0 0]
(C2) Sliding constraint		[node_no 2 0 0 0 0]
(C3) Spring constraint		[node_no 3 k_t k_r 0 0]
(C4) Hinged disk		[node_no 4 I_D k_r 0 0]
(C5) Sliding mass		[node_no 5 m k_t 0 0]
(C6) Mass and springs		[node_no 6 m k_t I k_r]

(C3) Spring constraint:

$$
\begin{aligned}
u_k(L_k) &= u_{k+1}(0), \quad \frac{d}{dx}u_k(L_k) = \frac{d}{dx}u_{k+1}(0) \\
EI_{k+1}\frac{d^2}{dx^2}u_{k+1}(0) &= EI_k\frac{d^2}{dx^2}u_k(L_k) + k_r\frac{d}{dx}u_k(L_k) \\
EI_{k+1}\frac{d^3}{dx^3}u_{k+1}(0) &= EI_k\frac{d^3}{dx^3}u_k(L_k) - k_t u_k(L_k)
\end{aligned}
\tag{14.77c}
$$

(C4) Hinged disk:

$$
\begin{aligned}
u_k(L_k) = 0, \quad u_{k+1}(0) = 0, \quad \frac{d}{dx}u_k(L_k) &= \frac{d}{dx}u_{k+1}(0) \\
EI_{k+1}\frac{d^2}{dx^2}u_{k+1}(0) &= EI_k\frac{d^2}{dx^2}u_k(L_k) + \left(k_r - I_D\omega^2\right)\frac{d}{dx}u_k(L_k)
\end{aligned}
\tag{14.77d}
$$

(C5) Sliding mass:

$$u_k(L_k) = u_{k+1}(0), \quad \frac{d}{dx}u_k(L_k) = 0, \quad \frac{d}{dx}u_{k+1}(0) = 0$$
$$EI_{k+1}\frac{d^3}{dx^3}u_{k+1}(0) = EI_k\frac{d^3}{dx^3}u_k(L_k) - \left(k_t - m\omega^2\right)u_k(L_k) \tag{14.77e}$$

(C6) Mass and Springs:

$$u_k(L_k) = u_{k+1}(0), \quad \frac{d}{dx}u_k(L_k) = \frac{d}{dx}u_{k+1}(0)$$
$$EI_{k+1}\frac{d^2}{dx^2}u_{k+1}(0) = EI_k\frac{d^2}{dx^2}u_k(L_k) + \left(k_r - I\omega^2\right)\frac{d}{dx}u_k(L_k) \tag{14.77f}$$
$$EI_{k+1}\frac{d^3}{dx^3}u_{k+1}(0) = EI_k\frac{d^3}{dx^3}u_k(L_k) - \left(k_t - m\omega^2\right)u_k(L_k)$$

Note that Eq. (14.77c) with vanishing spring coefficients describes the interconnection of S_k and S_{k+1} at Node k without an imposed constraint.

Boundary Conditions Seven types of boundary conditions of the stepped beam are specified at nodes 0 and N. These boundary conditions are the same as given in Table 14.2.1 if the bending stiffness of the beam is replaced by EI_1 and EI_N for $x = 0$ and L, respectively.

Eigenvalue Problem Equations (14.75), (14.77), and boundary conditions define an eigenvalue problem for the stepped beam, the solutions of which characterize the vibration of the beam in specific patterns or modes. Among many methods, the eigenvalue problem can be solved by the classical boundary value approach. To this end, assume the displacement of the kth beam segment as

$$u_k(x) = a_k \cos\beta_k x + b_k \sin\beta_k x + c_k\, e^{-\beta_k(L_k - x)} + d_k\, e^{-\beta_k x}, \quad 0 \le x \le L_k \tag{14.78}$$

where

$$\beta_k = \sqrt[4]{\frac{\rho_k\omega^2 - \mu_k}{EI_k}} \tag{14.79}$$

and a_k, b_k, c_k, and d_k are unknown coefficients to be determined. Substituting Eq. (14.78) into Eqs. (14.75), (14.77), and appropriate boundary conditions gives a set of algebraic equations

$$[D(\omega)]\{a\} = 0 \tag{14.80}$$

where $\{a\}$ is a vector of $4 \times N$ unknown coefficients

$$\{a\} = \begin{pmatrix} a_1 & b_1 & c_1 & d_1 & \cdots & a_N & b_N & c_N & d_N \end{pmatrix}^T$$

and $[D(\omega)]$ is a $4N$-by-$4N$ matrix with its elements being transcendental functions of ω. Thus, the characteristic equation is

$$\det[D(\omega)] = 0 \tag{14.81}$$

whose roots, $\omega_1, \omega_2, \ldots$, are the natural frequencies (eigenvalues) of the stepped beam. The mode shape (eigenfunction) associated with a natural frequency ω_n is determined in two steps:

(i) Solve the equation $[D(\omega_n)]\{a\} = 0$ for a nonzero vector $\{a\}$; and
(ii) Substitute $\{a\}$ into Eq. (14.78) to compute the mode shape of each beam segment.

Effects of Elastic Foundation on Mode Shapes Assume that the kth beam segment S_k is on an elastic foundation of coefficient μ_k. The mode shape $u_k(x)$ of S_k associated with a natural frequency ω takes one of the following three forms:

Case 1. $\rho_k\omega^2 > \mu_k$
By Eq. (14.76), $\beta_k^4 > 0$, and Eq. (14.78) gives a real expression of $u_k(x)$.

Case 2. $\rho_k\omega^2 = \mu_k$
By Eq. (14.76), $\beta_k^4 = 0$, and Eq. (14.75) indicates that the mode shape is

$$u_k(x) = a_k + b_k x + c_k x^2 + d_k x^3, \quad 0 \le x \le L_k \tag{14.82}$$

Case 3. $\rho_k\omega^2 > \mu_k$
By Eq. (14.76), $\beta_k^4 < 0$. In this case, the mode shape given by Eq. (14.78) is still valid, but becomes complex-valued. The equivalent real-valued mode shape is

$$\begin{aligned} u_k(x) = {} & e^{\gamma_k x}(a_k \cos \gamma_k x + b_k \sin \gamma_k x) \\ & + e^{-\gamma_k x}(c_k \cos \gamma_k x + d_k \sin \gamma_k x), \quad 0 \le x \le L_k \end{aligned} \tag{14.83}$$

where

$$\gamma_k = \frac{\sqrt{2}}{2}\sqrt[4]{\frac{\mu_k - \rho_k\omega^2}{EI_k}} \tag{14.84}$$

It is seen from the above cases that the value of μ_k may greatly affect the lower-frequency mode shapes.

EXAMPLE 4.1

In Fig. 14.4.2, a clamped-hinged two-span beam is constrained at its midpoint by translational spring. The equations for the eigenvalue problem of the beam are as follows:

Beam segments:

$$\frac{d^4u_k(x)}{dx^4} = \beta_k^4 u_k(x), \quad 0 \le x \le l, \quad k = 1, 2$$

where $\beta_k = \sqrt[4]{\rho_k\omega^2/EI_k}$

Boundary conditions:

$$u_1(0) = 0, \quad \frac{d}{dx}u_1(0) = 0$$

$$u_2(l) = 0, \quad \frac{d^2}{dx^2}u_2(l) = 0$$

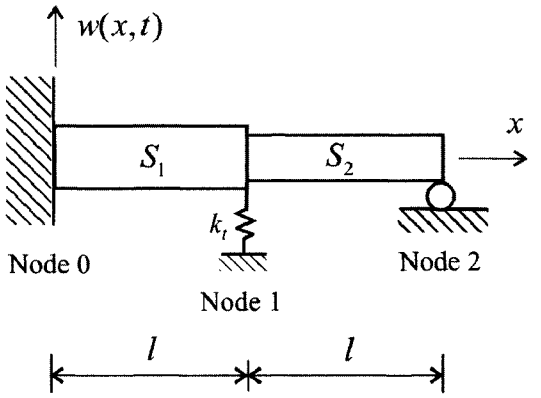


FIGURE 14.4.2 A clamped-hinged two-span beam.

Segment interconnection:

$$u_1(l) = u_2(0), \frac{d}{dx}u_1(l) = \frac{d}{dx}u_2(0)$$

$$EI_1\frac{d^2}{dx^2}u_1(l) = EI_2\frac{d^2}{dx^2}u_2(0)$$

$$EI_1\frac{d^3}{dx^3}u_1(l) = EI_2\frac{d^3}{dx^3}u_2(0) + k_t u_2(0)$$

Assume the displacements of the beam segments as

$$u_k(x) = A_k \cos\beta_k x + B_k \sin\beta_k x + C_k\, e^{-\beta_k(L_k - x)} + D_k\, e^{-\beta_k x}, \quad 0 \le x \le l$$

and substitute them in the equations of the eigenvalue problem, to obtain

$$[\Pi]\{U\} = 0$$

where

$$\{U\} = \begin{pmatrix} A_1 & B_1 & C_1 & D_1 & A_2 & B_2 & C_2 & D_2 \end{pmatrix}^T$$

$$[\Pi] = \begin{bmatrix} 1 & 0 & e_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & e_1 & -1 & 0 & 0 & 0 & 0 \\ c_1 & s_1 & 1 & e_1 & -1 & 0 & -e_2 & -1 \\ -\beta_1 s_1 & \beta_1 c_1 & \beta_1 & -\beta_1 e_1 & 0 & -\beta_2 & -\beta_2 e_2 & \beta_2 \\ -\beta_1^2 c_1 & -\beta_1^2 s_1 & \beta_1^2 & \beta_1^2 e_1 & \tau_{21}\beta_2^2 & 0 & -\tau_{21}\beta_2^2 e_2 & -\tau_{21}\beta_2^2 \\ \beta_1^3 s_1 & -\beta_1^3 c_1 & \beta_1^3 & -\beta_1^3 e_1 & -\dfrac{k_t}{EI_1} & \tau_{21}\beta_2^3 & -\tau_{21}\left(\beta_2^3 + \dfrac{k_t}{EI_2}\right)e_2 & \tau_{21}\left(\beta_2^3 - \dfrac{k_t}{EI_2}\right) \\ 0 & 0 & 0 & 0 & c_2 & s_1 & 1 & e_2 \\ 0 & 0 & 0 & 0 & -c_2 & -s_1 & 1 & e_2 \end{bmatrix}$$

with $c_j = \cos\beta_j l$, $s_j = \sin\beta_j l$, $e_j = e^{-\beta_j l}$, $j = 1, 2$, and $\tau_{21} = \frac{EI_2}{EI_1}$. The eigenvalues of the beam are the roots of the characteristic equation $\det[\Pi] = 0$.

14.4.2 Solution by the Toolbox

The Toolbox of this chapter offers MATLAB functions `setbeam3`, `plotmsh3`, and `animode3` for computation, illustration, and animation of modes of vibration of stepped beams; see Windows 4.1 to 4.3.

Window 4.1. Function setbeam3

MATLAB Function: **setbeam3**

Purpose:
To set up a stepped beam and to compute its eigensolutions.

Synopsis:

```
setbeam3(Beam_Spec, BC_Spec, Constr_Spec, n_mode)
```

Description:
`setbeam3(Beam_Spec, BC_Spec, Constr_Spec, n_mode)` specifies the parameters, boundary conditions, and constraints of a stepped beam by a matrix `Beam_Spec`, a vector `BC_Spec`, and a matrix `Constr_Spec`, respectively; and computes the first `n_mode` natural frequencies and mode shapes of the beam. By default, `n_mode = 8`. If the beam structure has no constraints, set `Constr_Spec = 0` or replace it by `[ ]` as a placeholder. For instance, `setbeam2(Beam_Spec, BC_Spec)` specifies a stepped beam without imposed constraints.

Specification of `Beam_Spec`, `BC_Spec`, and `Constr_Spec` follows this window.

Function `setbeam3` assigns the parameters, boundary conditions, and constraints for an N-segment stepped beam by the following arguments:

(i) The parameters of the beam segments are assigned by an N-row matrix

$$\texttt{Beam\_Spec} = \begin{bmatrix} \rho_1 & EI_1 & L_1 & \mu_1 \\ \rho_2 & EI_2 & L_2 & \mu_2 \\ \vdots & \vdots & \vdots & \vdots \\ \rho_N & EI_N & L_N & \mu_N \end{bmatrix} \tag{14.85a}$$

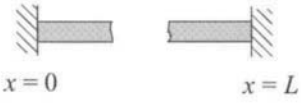
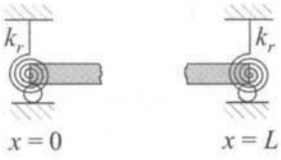
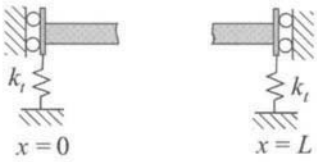
where the ith row contains the linear density, bending stiffness, length, and elastic foundation coefficient of the ith beam element. If none of the segments is constrained by elastic foundation,

$$\texttt{Beam\_Spec} = \begin{bmatrix} \rho_1 & EI_1 & L_1 \\ \rho_2 & EI_2 & L_2 \\ \vdots & \vdots & \vdots \\ \rho_N & EI_N & L_N \end{bmatrix} \tag{14.85b}$$

(ii) The boundary conditions of the beam are assigned with a row vector

$$\texttt{BC\_Spec} = \texttt{[BC\_Left BC\_Right]} \tag{14.86}$$

TABLE 14.4.2 Specification of Boundary Conditions for Stepped Beams

BC_Spec = [BC_Left BC_Right]	
(B1) Pinned or hinged end BC_Left = 1; BC_Right = 1	(B2) Clamped or fixed end  BC_Left = 2; BC_Right = 2
(B3) Free end BC_Left = 3; BC_Right = 3	(B4) Sliding or guided end BC_Left = 4; BC_Right = 4
(B5) Elastic-support BC_Left = [5 k_r k_t]; BC_Right = [5 k_r k_t]	(B6) Hinged-elastic end  BC_Left = [6 k_r]; BC_Right = [6 k_r]
(B7) Sliding-elastic end  BC_Left = [7 k_t]; BC_Right = [7 k_t]	(B0) End constraint (To be imposed according to Table 14.4.1) BC_Left = 0; BC_Right = 0

where `BC_Left` and `BC_Right` are two subvectors describing the boundary conditions at the left end (Node 0, $x = 0$) and the right end (Node N, $x = L$), respectively. Table 14.4.2 lists the entries of these subvectors for seven types of boundary conditions (B1 to B7). In addition, type (B0) is introduced for imposed constraints at Nodes 0 and N.

(iii) The constraints at the nodes of the beam are described by a matrix

$$\texttt{Constr\_Spec} = \begin{bmatrix} \texttt{Constr\_1} \\ \texttt{Constr\_2} \\ \vdots \\ \texttt{Constr\_Nc} \end{bmatrix} \tag{14.87}$$

where Nc is the number of constraints. The jth row of Constr_Spec specifies the jth constraint by

```
Constr_j = [node_no type_no par1 par2 ...]        (14.88)
```

where node_no is node number (0, 1, 2, ..., N), type_no is a constraint identification number, which is an integer from 1 to 6, and par1 par2 ... are any necessary parameters of the constraint. See Table 14.4.1 for the construction of Constr_j for six types of constraints.

Note: The information of a stepped beam that is set up by function setbeam3 can be accessed any time by calling function **systinfo3**, such as

```
>> systinfo3
```

Window 4.2. Function plotmsh3

MATLAB Function: **plotmsh3**

Purpose:
To plot mode shape of a stepped beam.

Synopsis:

```
plotmsh3(n)
plotmsh3(n, opt, npts)
[y, x, mdcoeff, ndpts]= plotmsh3(n, opt, npts)
```

Description:
plotmsh3(n)plots the nth mode shape (eigenfunction) of the beam over its length and displays the mathematical expressions of the eigenfunction.

plotmsh3(n, opt, npts)plots the nth mode shape of the beam with the options:

opt = 0	plot modal displacement
opt = 1	plot modal rotation (slope)
opt = 2	plot modal bending moment
opt = 3	plot modal shear force versus
opt = 4	plot modal displacement, rotation, bending moment, and shear force

Here npts is the number of equally-spaced points along each beam segment that are selected for computation. By default, opt = 0 and npts = 101.

[y, x, msh, ndpts] = plotmsh3(n, opt, npts) returns the modal distribution in a four-row matrix y, spatial points of computation in a vector x, the coefficients of the eigenfunction in a matrix msh, and nodal points of the mode shape in a vector ndpts. With x and y, the kth mode shape can be plotted later on by the MATLAB function plot as follows:

plot(x, y(1,:))	plot modal displacement
plot(x, y(2,:))	plot modal rotation (slope)
plot(x, y(3,:))	plot modal bending moment
plot(x, y(4,:))	plot modal shear force.

Window 4.2. Function plotmsh3 (*Continued*)

For an N-segment beam, matrix msh has N rows. The kth row of msh contains the coefficients of the mode shape $u_k(x)$ of the kth beam segment as follows

$$\text{msh} = [\beta \quad a_k \quad b_k \quad c_k \quad d_k]$$

where $\beta = \beta_k$ for the mode shape given by Eq. (14.78); $\beta = 0$ for the mode shape given by Eq. (14.82); $\beta = -\gamma_k$ for the mode shape given by Eq. (14.83); and a_k, b_k, c_k, and d_k are the constants in one of the above-mentioned equations.

Window 4.3. Function animode3

MATLAB Function: **animode3**

Purpose:
To play animated displacement of a stepped beam in a mode of vibration.

Synopsis:

```
animode3(k)
animode3(k, IS_Control, tf, n_frames)
F = animode3(k, IS_Control, tf, n_frames)
```

Description:
`animode3(k)` animates the kth mode of vibration of the beam.

`animode3(k, IS_Control, tf, n_frames)` plays the animated vibration of the beam in the kth mode in a specified manner, where `IS_control` is an animation control parameter with the options:

```
IS_control = 0    play frames of continuously
IS_control = 1    play frames one by one
```

`tf` is total animation time, and `n_frames` is the number of frames of animation. By default, `IS_control = 0`, `tf = 8*T` with T being the period of the mode, and `n_frames = 97`. Parameters `IS_Control`, `tf`, and `n_frames` can be replaced by [ ] as a placeholder for their default setup.

`F = animode3(k, IS_Control, tf, n_frames)` returns a set of frames of animated beam vibration in a matrix F, which can be played by the MATLAB function `movie(F)`.

EXAMPLE 4.2

In Fig. 14.4.3, a three-segment beam is hinged and constrained by a torsional spring at the left end, and has a mounted lumped mass at the right end. The middle beam segment

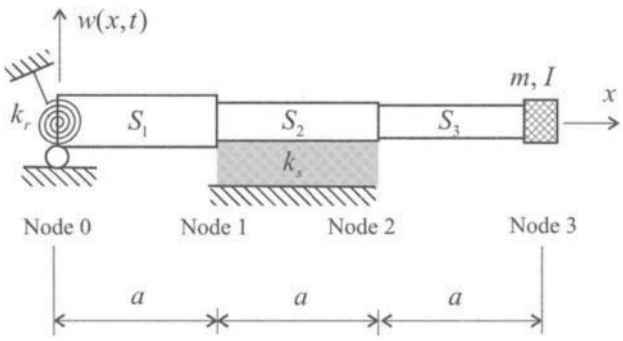


FIGURE 14.4.3 A three-segment beam.

is constrained by an elastic foundation of coefficient k_s. The parameters of the beam structure are as follows:

Segment 1 (S_1): $\rho_1 = 1, \quad EI_1 = 40, \quad L_1 = a = 0.5$

Segment 2 (S_2): $\rho_2 = 0.7, \quad EI_1 = 30, \quad L_2 = a = 0.5, \quad k_s = 500$

Segment 3 (S_3): $\rho_3 = 0.5, \quad EI_3 = 20, \quad L_3 = a = 0.5$

Left end: $k_r = 25$

Right end: $m = 0.2, \quad I = 0.125$

The commands

```
>> Beam_Spec = [1 40 0.5 0; 0.7 30 0.5 500; 0.5 20 0.5 0];
>> Constr_Spec = [3 6 0.2 0 0.125 0];
>> setbeam3(Beam_Spec, [6 25 0], Constr_Spec)
```

yield the first eight natural frequencies of the stepped beam as shown below, and plot the first four mode shapes in Fig. 14.4.4.

```
First 8 Eigenvalues
-------------------
  wn, natural frequencies in rad/sec
  fn = wn/(2*pi), natural frequencies in Hertz

     wn      fn
  1.0e+003 *

  0.0090  0.0014
  0.0209  0.0033
  0.0608  0.0097
  0.1584  0.0252
  0.3156  0.0502
  0.5271  0.0839
  0.7914  0.1260
  1.1183  0.1780

>> plotmsh3(2,4)
```

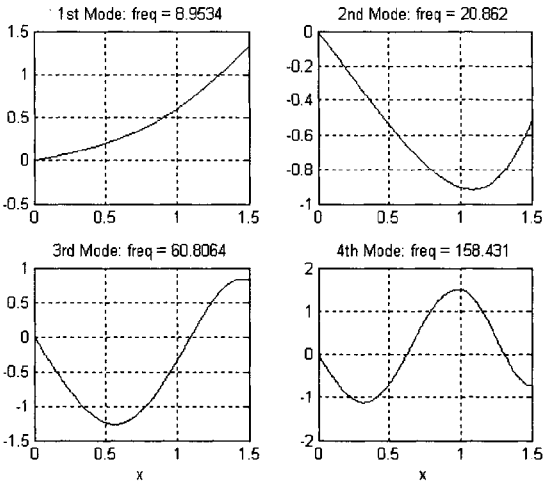


FIGURE 14.4.4 The first four mode shapes of the beam.

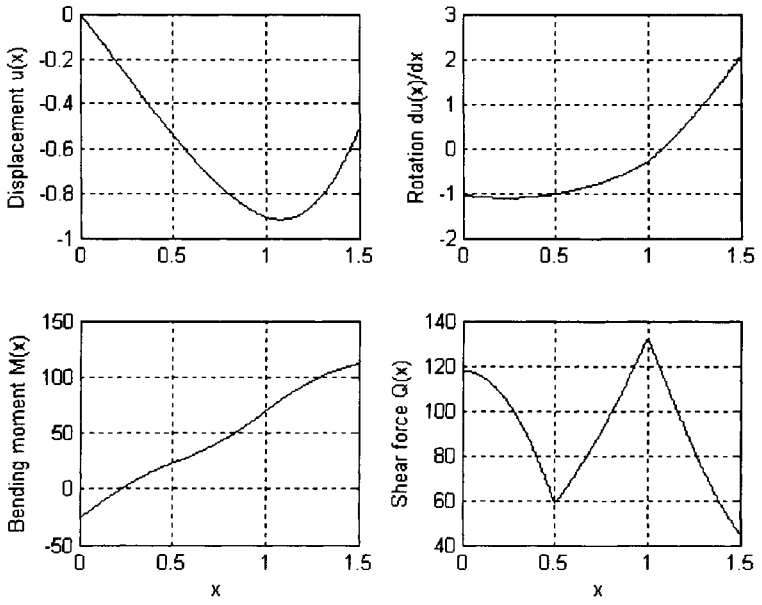


FIGURE 14.4.5 Modal distribution of the second mode.

plots the spatial distributions of the modal displacement, slope, bending moment, and shear force of the second mode in Fig. 14.4.5, and gives the mathematical expressions of the modal displacement of each beam segment in the second mode as follows.

```
Mode 2: w2 = 20.862 rad/s
------------------------

Flexible mode shape:

Segment 1, 0<=x<=L1
  u(x) = a1*cos(beta*x) + a2*sin(beta*x) + a3*exp(-beta*(L1-x))
         + a4*exp(-beta*x),
```

```
  with beta = 1.8162, a1 = -0.97929, a2 = -0.63597, a3 = 0.6724,
              a4 = -0.19753, L1 = 0.5
Segment 2, 0<=x<=L2 (effect of elastic foundation, rou*w2^2<ks)
  u(x) = exp(beta*x)*(a1*cos(beta*x)+a2*sin(beta*x))
         + exp(-beta*x)*(a3*cos(beta*x)+a4*sin(beta*x))
  with beta = 1.1295, a1 = -0.97929, a2 = -0.63597, a3 = 0.6724,
              a4 = -0.19753, L2 = 0.5
Segment 3, 0<=x<=L3
  u(x) = a1*cos(beta*x) + a2*sin(beta*x) + a3*exp(-beta*(L3-x))
         + a4*exp(-beta*x),
  with beta = 1.8162, a1 = -0.97929, a2 = -0.63597, a3 = 0.6724,
              a4 = -0.19753, L3 = 0.5
```

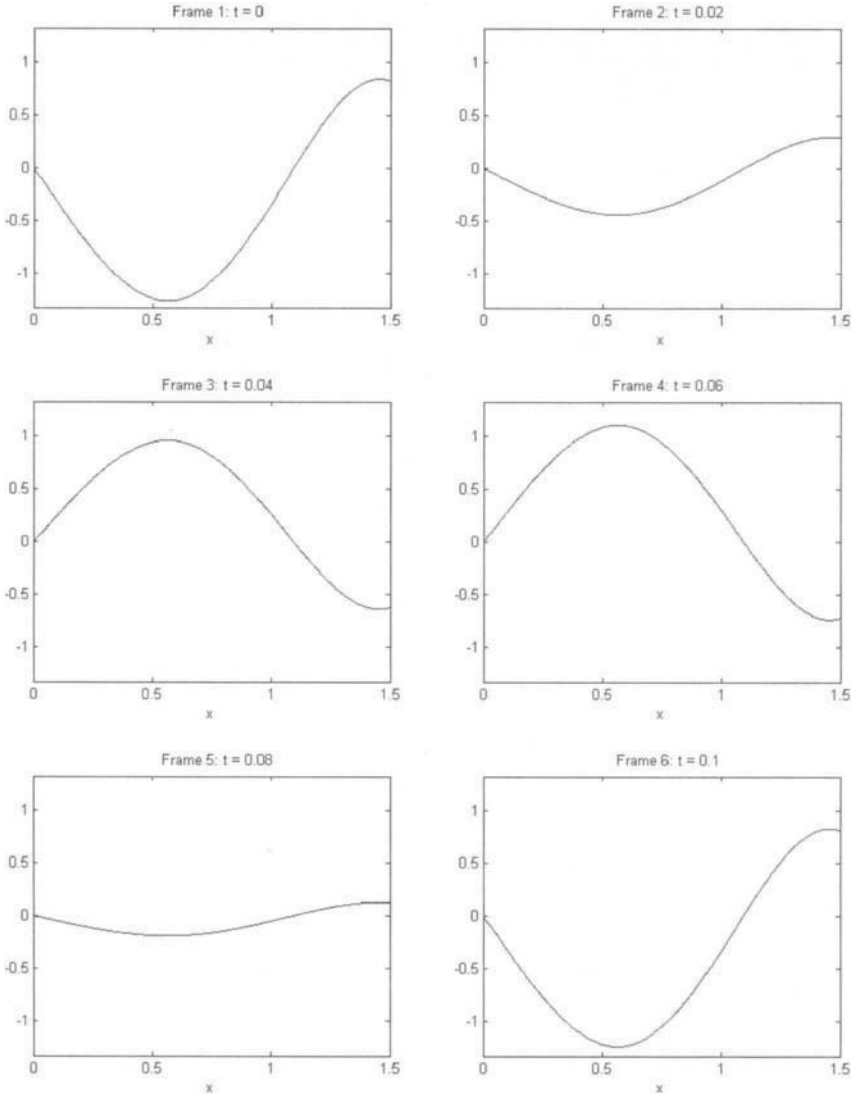


FIGURE 14.4.6 Frames of the animated beam vibration in the third mode.

The effect of the elastic foundation on the distribution of the modal shear force of the segment S_2 is seen from the figure.

In addition,

```
>> animode3(3, 1, 0.1, 6)
```

plays the animated free vibration of the beam in the third mode, see Fig. 14.4.6.

EXAMPLE 4.3

Consider the beam in Example 2.7, which has a hinge constraint at its midpoint.

```
>> Beam_Spec = [0.3 40 1; 0.3 40 0.5; 0.3 40 0.5];
>> Constr_Spec = [1 1 0 0 0 0; 2 6 6 0 0 0];
>> setbeam3(Beam_Spec, [1 1], Constr_Spec)
```

gives the first eight natural frequencies of the beam as below and the first four mode shapes in Fig. 14.4.7. It is seen that the results (approximate solutions) in Example 2.7 are in agreement with what are obtained here.

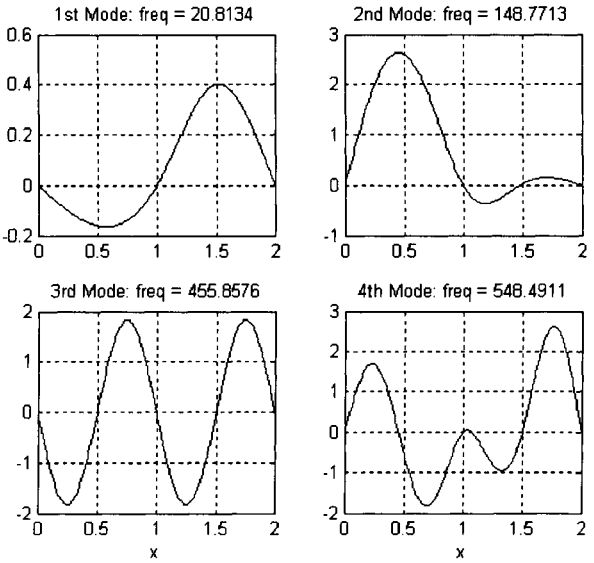


FIGURE 14.4.7

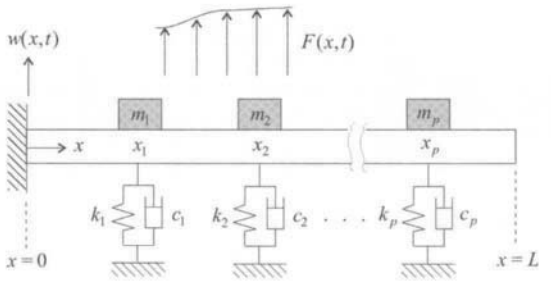
```
First 8 Eigevalues
------------------
  wn, natural frequencies in rad/sec
  fn = wn/(2*pi), natural frequencies in Hertz
     wn     fn
  1.0e+003 *
  0.0208   0.0033
  0.1488   0.0237
```

```
0.4559   0.0726
0.5485   0.0873
0.8028   0.1278
1.1381   0.1811
1.8234   0.2902
2.0080   0.3196
```

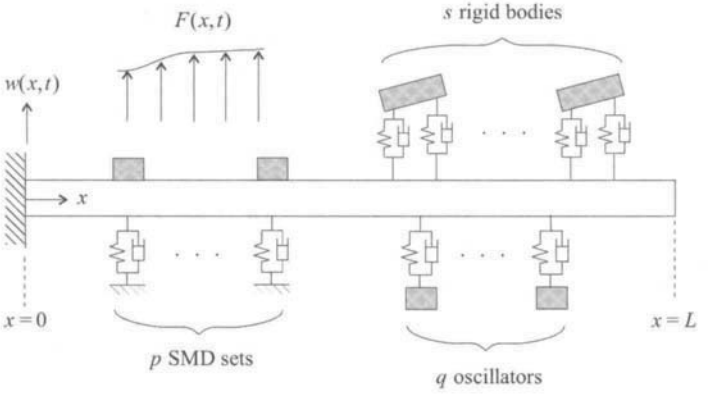
14.5 Quick Solution Guide

Beam Structures in Consideration

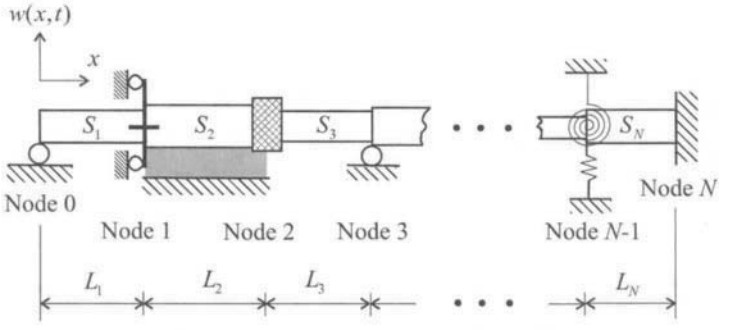
Constrained Beams



Combined Beams



Stepped Beams



Problems to Solve

(a) Eigenvalue problem
(b) Transient response
(c) Frequency response

MATLAB Functions

This chapter has a set (toolbox) of MATLAB functions for dyanmic analysis of constrained, combined and stepped beams; see the table below. Refer to the corresponding window or section for the utility of each function.

	Constrained Beams	
`setbeam`	Set up a constrained beam and compute its eigensolutions	Window 2.1
`systinfo`	Display system information and eigensolutions	Section 14.2
`plotmsh`	Plot mode shape distribution along beam length	Window 2.2
`animode`	Play animated beam response in a mode of vibration	Window 2.3
`eiglocus`	Plot eigenvalue loci of a beam with an added spring-mass-damper set	Window 2.4
`frfbeam`	Plot the steady-state response of a constrained beam subject to a pointwise sinusoidal force	Window 2.5
	Combined Beams	
`setbeam2`	Set up a combined beam and compute its eigensolutions	Window 3.1
`systinfo`	Display system information and eigensolutions	Section 14.3
`plotmsh`	Plot mode shape distribution along beam length	Window 2.2
`animode`	Play beam response in a mode of vibration	Window 2.3
`trbeam2`	Plot the time history of the transient displacement of a combined beam subject to external and initial disturbances	Window 3.2
`trbeam2s`	Plot the spatial distribution of the displacement of a combined beam subject to external and initial disturbances	Window 3.3
`beammovie2`	Animate transient vibration of a combined beam	Window 3.4
`frfbeam2`	Plot the steady-state response of a combined beam subject to a pointwise sinusoidal force	Window 3.5
	Stepped Beams	
`setbeam3`	Set up a stepped beam, and compute its eigensolutions	Window 4.1
`systinfo3`	Display system information and eigensolutions	Section 14.4
`plotmsh3`	Plot mode shape distribution along beam length	Window 4.2
`animode3`	Play animated beam response in a mode of vibration	Window 4.3
	Utilities	
`TBdemo`	To show how the Toolbox works and what it can do	Section 14.1
`TBinfo`	To show the information of the Toolbox	Section 14.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 14.1

Solution Procedure

Dynamic analysis by the Toolbox tables are shown in the following two steps.

Step 1. Set up system parameters, boundary conditions, and damping
For a constrained beam, type

```
>> setbeam(rou, EI, L, Dmp_Spec, BC_Spec, SMD_Spec)
```

For a combined beam, type

```
>> setbeam2(rou, EI, L, Dmp_Spec, BC_Spec, SMD_Spec,
   Osc_Spec, Rbd_Spec)
```

For a stepped beam, type

```
>> setbeam3(Beam_Spec, BC_Spec, Constr_Spec)
```

Functions `setbeam`, `setbeam2`, and `setbeam3` also yield the eigensolutions of the structures in consideration.

Step 2. Plot and animate dynamic response

For the transient response of a constrained or combined beam subject to an external load and initial disturbances at location `x = xr`, type

```
>> trfbeam2(xr, Load_Spec, IC)
```

To animate the *k*th mode of vibration of a constrained or combined beam, type

```
>> animode(k)
```

To animate the *k*th mode of vibration of a stepped beam, type

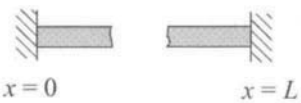
```
>> animode3(k)
```

To animate the transient vibration of a constrained or combined beam, type

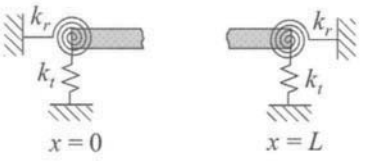
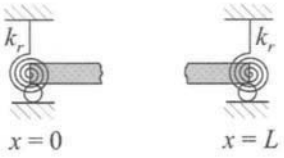
```
>> beammovie2(Load_Spec, IC)
```

The specification of `BC_Spec`, `Dmp_Spec`, `SMD_Spec`, `Osc_Spec`, `Rbd_Spec`, `Load_Spec`, `IC`, `Beam_Spec`, and `Constr_Spec` is given as follows.

(1) Specification of vector `BC_Spec` for beam boundary conditions (as in Tables 14.2.3 and 14.4.2)

`BC_Spec = [BC_Left BC_Right]`	
(B1) Pinned or hinged end `BC_Left = 1; BC_Right = 1`	(B2) Clamped or fixed end  `BC_Left = 2; BC_Right = 2`
(B3) Free end `BC_Left = 3; BC_Right = 3`	(B4) Sliding or guided end `BC_Left = 4; BC_Right = 4`

(continued)

BC_Spec = [BC_Left BC_Right]	
(B5) Elastic-support  BC_Left = [5 k_r k_t]; BC_Right = [5 k_r k_t]	(B6) Hinged-elastic end  BC_Left = [6 k_r]; BC_Right = [6 k_r]
(B7) Sliding-elastic end BC_Left = [7 k_t]; BC_Right = [7 k_t]	(B0) End constraint (For stepped beams only) BC_Left = 0 BC_Right = 0

(2) Specification of vector Dmp_Spec for damping in a constrained/combined beam

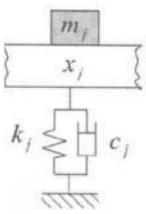
Option 1: Viscous damping. Dmp_Spec = dv where dv is the viscous damping coefficient.

Option 2: Modal damping. Dmp_Spec is a vector of m+1 parameters

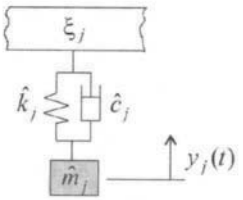
```
Dmp_Spec = [1 zeta_1 zeta_2 ... zeta_m ]
```

where zeta_1, zeta_2, ... zeta_m are modal damping ratios, whose values fall between 0 and 1. If m is less than the number n of the terms in Eq. (14.5), the modes of numbers $m, m+1, \ldots, n$ will have the same damping ratio as zeta_m.

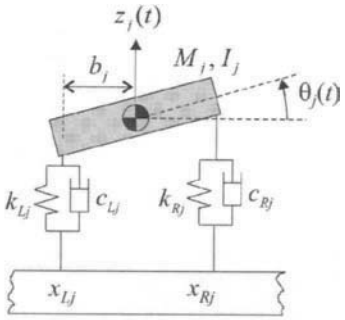
Option 3: No damping. Dmp_Spec = [] or 0.



SMD set



Oscillator



Rigid body

(3) Specification of matrix SMD_Spec for attached SMDs

$$\texttt{SMD\_Spec} = \begin{bmatrix} x_1 & k_1 & m_1 & c_1 \\ x_2 & k_2 & m_2 & c_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_p & k_p & m_p & c_p \end{bmatrix}$$

where the jth row of the matrix specifies the jth SMD located at $x = x_j$.

(4) Specification of matrix Osc_Spec for oscillators

$$\texttt{Osc\_Spec} = \begin{bmatrix} \xi_1 & \hat{k}_1 & \hat{m}_1 & \hat{c}_1 \\ \xi_2 & \hat{k}_2 & \hat{m}_2 & \hat{c}_2 \\ \vdots & \vdots & \vdots & \vdots \\ \xi_q & \hat{k}_q & \hat{m}_q & \hat{c}_q \end{bmatrix}$$

where the jth row of the matrix specifies the jth oscillator located at $x = \xi_j$.

(5) Specification of matrix Rbd_Spec for rigid bodies

$$\texttt{Rbd\_Spec} = \begin{bmatrix} M_1 & I_1 & b_1 & x_{L1} & k_{L1} & c_{L1} & x_{R1} & k_{L1} & c_{L1} \\ M_2 & I_2 & b_2 & x_{L2} & k_{Ls} & c_{L2} & x_{R2} & k_{L2} & c_{L2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M_s & I_s & b_s & x_{Ls} & k_{Ls} & c_{Ls} & x_{Rs} & k_{Ls} & c_{Ls} \end{bmatrix}$$

where the jth row of the matrix specifies the jth rigid body mounted at points x_{Lj} and x_{Rj}, with M_j and I_j being the mass and moment of inertia, respectively, and b_j being the distance between the left supporting point and the mass center of the rigid body.

(6) Specification of external loads by vector Load_Spec (as in Table 14.3.1)

External Load	*F(x,t)*	Load_Spec
(L0) Pointwise impulse load	$I_0\delta(t)\delta(x - x_f)$	[0 x_f I_0]
	$I_0\delta(t - T_d)\delta(x - x_f)$	[0 x_f I_0 T_d]
(L1) Pointwise step load	$F_0\delta(x - x_f)$	[1 x_f F_0]
	$F_0u(t - T_d)\delta(x - x_f)$	[1 x_f F_0 T_d]
(L2) Pointwise sinusoidal load	$A\sin(\omega t + \phi_0)\delta(x - x_f)$	[2 x_f A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
	$A\sin(\omega t + \phi_0)u(t - T_d)\delta(x - x_f)$	[2 x_f A ω ϕ_0 T_d] ω in rad/s, ϕ_0 in degrees
(L10) Uniformly distributed impulse load	$I_0\delta(t)$, for $x_1 \le x \le x_1$	[10 x_1 x_2 I_0]
	$I_0\delta(t - T_d)$, for $x_1 \le x \le x_1$	[10 x_1 x_2 I_0 T_d]
(L11) Uniformly distributed step load	F_0, for $x_1 \le x \le x_1$	[11 x_1 x_2 F_0]
	$F_0u(t - T_d)$, for $x_1 \le x \le x_1$	[11 x_1 x_2 F_0 T_d]

(continued)

External Load	*F(x,t)*	Load_Spec
(L12) Uniformly distributed sinusoidal load	$A\sin(\omega t+\phi_0)$, for $x_1 \le x \le x_1$	[12 x_1 x_2 A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
	$A\sin(\omega t+\phi_0)u(t-T_d)$, for $x_1 \le x \le x_1$	[12 x_1 x_2 A ω ϕ_0 T_d] ω in rad/s, ϕ_0 in degrees
(L20) Pointwise impulse torque	$-I_0\delta(t)\dfrac{d\delta(x-x_\tau)}{dx}$	[20 x_τ I_0]
	$-I_0\delta(t-T_d)\dfrac{d\delta(x-x_\tau)}{dx}$	[20 x_τ I_0 T_d]
(L21) Pointwise step torque	$-\tau_0\dfrac{d\delta(x-x_\tau)}{dx}$	[21 x_τ τ_0]
	$-\tau_0\dfrac{d}{dx}\delta(x-x_\tau)u(t-T_d)$	[21 x_τ τ_0 T_d]
(L22) Pointwise sinusoidal torque	$-A\sin(\omega t+\phi_0)\dfrac{d\delta(x-x_\tau)}{dx}$	[22 x_τ A ω ϕ_0] ω in rad/s, ϕ_0 in degrees
	$-A\sin(\omega t+\phi_0)\dfrac{d}{dx}\delta(x-x_\tau)u(t-T_d)$	[22 x_τ A ω ϕ_0 T_d] ω in rad/s, ϕ_0 in degrees

In the previous table, $u(t-T_d)$ is the delayed unit step function, which is one for $t > T_d$, and zero otherwise, and T_d is a nonnegative time delay.

(7) Specification of initial disturbances by matrix IC
Let the initial conditions of a constrained or combined beam be

$$w(x,0)=a_0(x), \quad \frac{\partial}{\partial t}w(x,0)=b_0(x), \quad 0 \le x \le L$$

where L is the beam length. In the Toolbox, these initial disturbances are specified by a 2-by-np matrix

$$\texttt{IC}=\begin{bmatrix} a_0(x_1) & a_0(x_2) & \cdots & a_0(x_{np}) \\ b_0(x_1) & b_0(x_2) & \cdots & b_0(x_{np}) \end{bmatrix}$$

where $a_0(x_j)$ and $b_0(x_j)$ are the values of the initial displacement and velocity at np equally spaced points along the beam length $x_j = j \cdot L/np, j = 1,2,\ldots,np$.

(8) Specification of beam segments of a stepped beam by matrix Beam_Spec

$$\texttt{Beam\_Spec}=\begin{bmatrix} \rho_1 & EI_1 & L_1 & \mu_1 \\ \rho_2 & EI_2 & L_2 & \mu_2 \\ \vdots & \vdots & \vdots & \vdots \\ \rho_N & EI_N & L_N & \mu_N \end{bmatrix}$$

where the ith row contains the linear density, bending stiffness, length, and elastic foundation coefficient of the ith beam element. If none of the segments is constrained

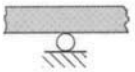
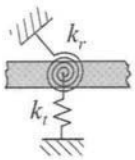
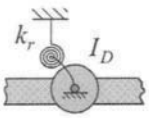
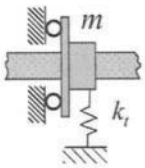
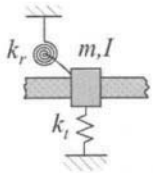
by elastic foundation,

$$\texttt{Beam\_Spec} = \begin{bmatrix} \rho_1 & EI_1 & L_1 \\ \rho_2 & EI_2 & L_2 \\ \vdots & \vdots & \vdots \\ \rho_N & EI_N & L_N \end{bmatrix}$$

(9) Specification of constraints imposed on a stepped beam by matrix `Constr_Spec`

$$\texttt{Constr\_Spec} = \begin{bmatrix} \texttt{Constr\_1} \\ \texttt{Constr\_2} \\ \vdots \\ \texttt{Constr\_Nc} \end{bmatrix}$$

where the jth row of `Constr_Spec` specifies the jth constraint, and is given in the following table (same as Table 14.4.1).

Constraint Type		`Constr_j`
(C1) Hinge		[node_no 1 0 0 0 0]
(C2) Sliding constraint		[node_no 2 0 0 0 0]
(C3) Spring constraint		[node_no 3 k_t k_r 0 0]
(C4) Hinged disk		[node_no 4 I_D k_r 0 0]
(C5) Sliding mass		[node_no 5 m k_t 0 0]
(C6) Mass and springs		[node_no 6 m k_t I k_r]

14.6 References

1. Den Hartog J P 1985 *Mechanical Vibrations*, Dover Publications: New York.
2. Inman D J 2000 *Engineering Vibrations*, 2nd ed., Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
3. Meirovitch L 1967 *Analytical Methods in Vibrations*, Macmillan: New York.
4. Rao S S 2003 *Mechanical Vibrations*, 4th ed., Prentice-Hall, Inc.: Upper Saddle River, New Jersey.
5. Timoshenko S 1990 *Vibration Problems in Engineering*, 5th ed., John Wiley & Sons.
6. Caughey T K, O'Kelley M E J 1965 "Classical Normal Modes in Damped Linear Dynamic Systems," *ASME Journal of Applied Mechanics*, Vol. 49: pp. 867–870.

PART V
Two-Dimensional Elastic Continua

This Page Intentionally Left Blank

15 Static Analysis of Linearly Elastic Bodies

Inside
- Getting Started
- Theory of Linear Elasticity
- Elasticity Problems in Two Dimensions
- Finite Element Method for 2-D Elasticity Problems
- Quick Solution Guide
- References

15.1 Getting Started

What Is in this Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for static analysis of linearly elastic bodies in two and three dimensions.

System Requirements for the MATLAB Toolbox

- PC with Win 98/NT/2000 and XP, or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **`TBdemo`** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox has been properly installed, and it is ready for use.

[1] If the M-files of the toolbox are put in the MATLAB work folder, there is no need to set a path.

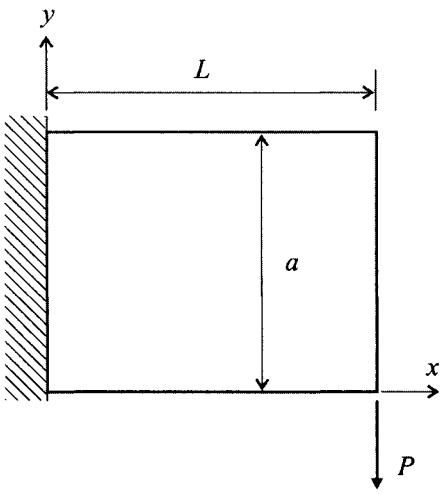


FIGURE 15.1.1 A rectangular elastic body under a pointwise load.

Quick Tutorial

To show how to use the MATLAB Toolbox, consider a rectangular cantilever plate in Fig. 15.1.1, with length $L = 3$ m, height $a = 2$ m, thickness $t = 0.1$ m, Young's modulus $E = 50{,}000$ Pa, and Poisson's ratio $\mu = 0.3$. The plate is subject to a pointwise load $P = 50$ N at its lower-right corner. The solution of this elasticity problem is obtained numerically by typing the following in the MATLAB command window:

```
>> E = 5e4; mu = 0.3; L = 3; a = 2; t = 0.1;
>> BC_Spec = [0 0 0 1];
>> setrectan([E mu L a t], BC_Spec)
>> Load_Spec = [5 0 -50];
>> resprectan(0, Load_Spec)
```

where >> is the prompt in the MATLAB command window. This yields a deformed finite element mesh of the body plotted in Fig. 15.1.2, and the displacements at the nodes $1, 2, \ldots, 25$ shown below

```
Nodal Displacements:

  Node       u        v
 --------------------------
  1.0000      0        0
  2.0000  -0.0241  -0.0220
  3.0000  -0.0422  -0.0536
  4.0000  -0.0546  -0.1010
  5.0000  -0.0597  -0.1638
  6.0000      0        0
  7.0000  -0.0101  -0.0170
  8.0000  -0.0183  -0.0532
  9.0000  -0.0225  -0.0981
 10.0000  -0.0279  -0.1529
 11.0000      0        0
```

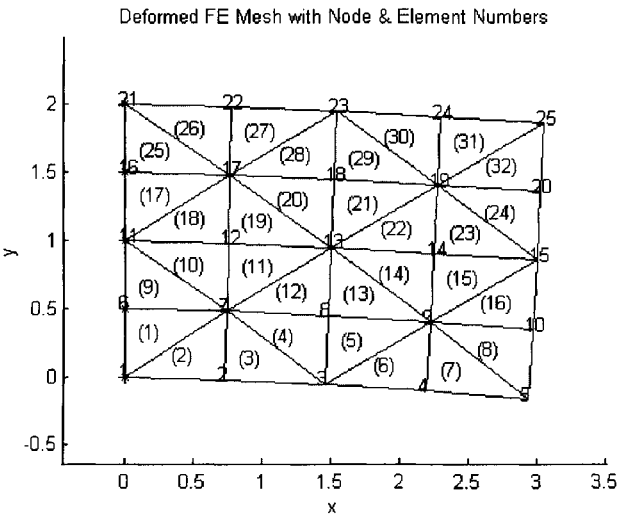


FIGURE 15.1.2 A finite element mesh of the deformed body.

```
12.0000    0.0002   -0.0184
13.0000    0.0006   -0.0515
14.0000    0.0008   -0.0974
15.0000    0.0009   -0.1448
16.0000         0         0
17.0000    0.0101   -0.0174
18.0000    0.0187   -0.0535
19.0000    0.0241   -0.0966
20.0000    0.0247   -0.1419
21.0000         0         0
22.0000    0.0239   -0.0223
23.0000    0.0408   -0.0544
24.0000    0.0493   -0.0977
25.0000    0.0521   -0.1412
```

Accordingly, the displacements of the body at the lower-right corner are $u = -0.0597$ m (in the x-direction), and $v = -0.1638$ m (in the y-direction).

In the above static analysis of the elastic body, two MATLAB functions have been used:

(a) Function **setrectan** that inputs the material and geometric parameters of the body, assigns the boundary conditions, and sets up a finite element mesh for computation; and

(b) Function **resprectan** that, by the finite element method, computes the static response of the elastic body under a pointwise load and displays the computed results.

These functions and others from the Toolbox will be introduced in the subsequent sections.

For a quick solution, the user may refer directly to the Quick Solution Guide (Section 15.5), or get the information on the Toolbox by typing **TBinfo** in the MATLAB command window. To run the examples contained in this chapter, type **RunEx** in the MATLAB command window.

To better understand how the toolbox works, the user is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 15.6.

15.2 Theory of Linear Elasticity

15.2.1 Stress and Strain

Stress Components Stress is a measure of the intensity of force. The unit of stress is force per unit area, which is Pa (N/m^2) in the standard international system or psi (lb/in^2) in the U.S. customary system. The stress at a point of a three-dimensional elastic body is defined by a cubic element in the rectangular frame $Oxyz$; see Fig. 15.2.1, where σ_x, σ_y, and σ_z are normal stresses and τ_{xy}, τ_{yx}, τ_{yz}, τ_{zy}, τ_{zx}, τ_{xz} are shear stresses. For static equilibrium, the following conditions about shear stresses hold true:

$$\tau_{xy} = \tau_{yx}, \quad \tau_{yz} = \tau_{zy}, \quad \tau_{zx} = \tau_{xz} \tag{15.1}$$

So, the stress at a point is described by six components (σ_x, σ_y, σ_z, τ_{xy}, τ_{xz}, τ_{yz}).

Consider an inclined plane ABC of unit normal $\boldsymbol{N} = l\boldsymbol{i} + m\boldsymbol{j} + n\boldsymbol{k}$ in Fig. 15.2.2, where $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ are the unit vectors in the x, y, and z directions, respectively; and l, m, and n are the direction cosines, i.e., $l = \cos(\boldsymbol{N}, x), m = \cos(\boldsymbol{N}, y), n = \cos(\boldsymbol{N}, z)$. The direction cosines are related by $l^2 + m^2 + n^2 = 1$. Let P_x, P_y, and P_z be the components of the resultant stress $\boldsymbol{P}$ acting on the plane ABC. The force balance on the tetrahedron $OABC$ leads to the following equilibrium equations

$$\begin{aligned} P_x &= \sigma_x l + \tau_{xy} m + \tau_{xz} n \\ P_y &= \tau_{xy} l + \sigma_y m + \tau_{yz} n \\ P_z &= \tau_{xz} l + \tau_{yz} m + \sigma_z n \end{aligned} \tag{15.2}$$

Furthermore, decompose the resultant stress $\boldsymbol{P}$ into a component σ_N that is normal to the inclined plane ABC (parallel to the normal $\boldsymbol{N}$) and a component τ that is tangential to the plane

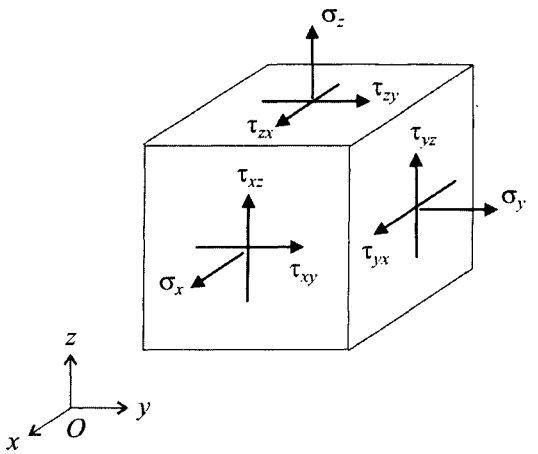


FIGURE 15.2.1 Stress components on a cubic element.

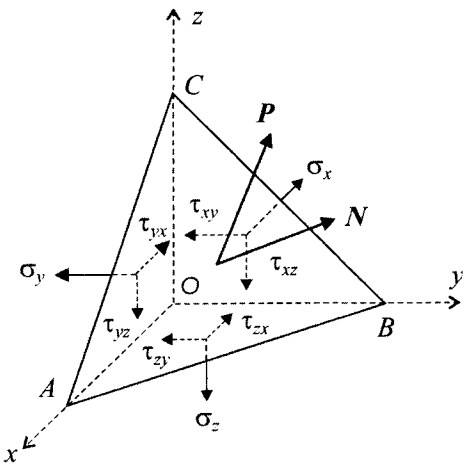


FIGURE 15.2.2 Resultant stress acting on an inclined plane.

(perpendicular to N). The normal and shear stresses on the plane are given by

$$\sigma_N = \sigma_x l^2 + \sigma_y m^2 + \sigma_z n^2 + 2(\tau_{xy} lm + \tau_{xz} ln + \tau_{yz} mn)$$
$$\tau = \left\{(\sigma_x l + \tau_{xy} m + \tau_{xz} n)^2 + (\tau_{xy} l + \sigma_y m + \tau_{yz} n)^2 + (\tau_{xz} l + \tau_{yz} m + \sigma_z n)^2 - \sigma_N^2\right\}^{1/2} \tag{15.3}$$

Deformation In Fig. 15.2.3, a body B undergoes a deformation. Consider a point P of the undeformed body B that is defined by the position vector $\boldsymbol{r} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$, where $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$ are the unit vectors of the Cartesian coordinate axes. The deformation at P is described by the vector

$$\boldsymbol{U} = u\boldsymbol{i} + v\boldsymbol{j} + w\boldsymbol{k} \tag{15.4}$$

where u, v, and w are the components of the displacement. Due to the deformation, point P moves to point $\mathscr{P}$, with the position vector $\boldsymbol{r} + \boldsymbol{U}$, and the deformed body occupies the region $\mathscr{B}$.

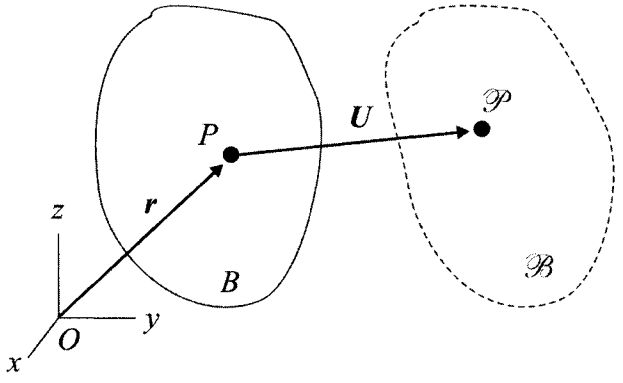


FIGURE 15.2.3 A body undergoing a deformation.

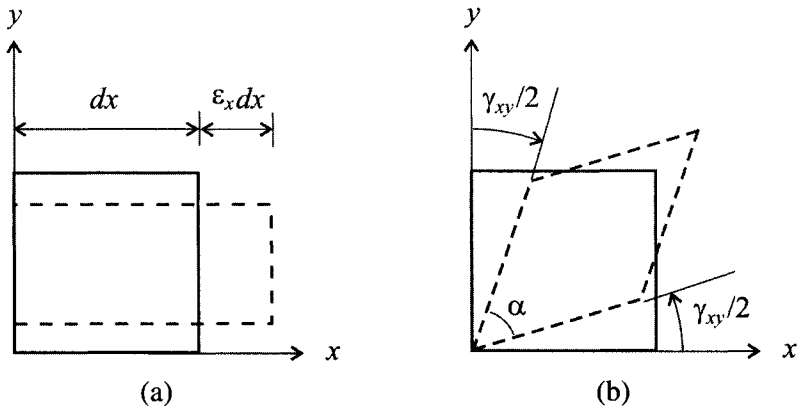


FIGURE 15.2.4 Normal and shear strain components.

Strain Components Strain is a measure of the intensity of deformation. The strain at a point is described by normal strain components $\varepsilon_x, \varepsilon_y$, and ε_z, and shear strain components $\gamma_{xy}, \gamma_{yx}, \gamma_{xz}, \gamma_{zx}, \gamma_{yz}, \gamma_{zy}$. A normal strain component, say ε_x, is the unit elongation of an infinitesimal element in the x-direction; see Fig. 15.2.4(a). A shear strain component, say γ_{xy}, is the change of the angle between two adjacent sides of an infinitesimal element that are originally in the x- and y- directions; see Fig. 15.2.4(b). By definition,

$$\gamma_{xy} = \gamma_{yx}, \quad \gamma_{yz} = \gamma_{zy}, \quad \gamma_{zx} = \gamma_{xz} \tag{15.5}$$

Thus, the strain at a point is described by six components $(\varepsilon_x, \varepsilon_y, \varepsilon_z, \gamma_{xy}, \gamma_{xz}, \gamma_{yz})$. Strains are dimensionless quantities, whose units are micrometers per meter (mm/m), inches per inch (in/in), and μ (10^{-6}). The strains of most engineering materials are normally below 0.002 or 2000 μ.

Strain at a point is related to the derivatives of the deformation in the neighborhood of the point; see the strain-displacement relations in Section 15.2.2.

Transformation of Stress and Strain Define the matrix of stress components associated with frame $Oxyz$ as

$$[\sigma] = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{bmatrix} \tag{15.6}$$

which describes a stress tensor. The stress components in another frame $Ox'y'z'$ are determined by the standard tensor transformation

$$[\sigma'] = \begin{bmatrix} \sigma_{x'} & \tau_{x'y'} & \tau_{x'z'} \\ \tau_{x'y'} & \sigma_{y'} & \tau_{y'z'} \\ \tau_{x'z'} & \tau_{y'z'} & \sigma_{z'} \end{bmatrix} = [T]^T [\sigma] [T] \tag{15.7}$$

where $[T]$ is a matrix of direction cosines

$$[T] = \begin{bmatrix} \cos(x', x) & \cos(y', x) & \cos(z', x) \\ \cos(x', y) & \cos(y', y) & \cos(z', y) \\ \cos(x', z) & \cos(y', z) & \cos(z', z) \end{bmatrix} \tag{15.8}$$

with $\cos(i', j)$ being the cosine of the angle between the i' and j axes. The $[T]$ is an orthogonal matrix; namely, $[T]^T [T] = [T][T]^T = [I]$, where $[I]$ is the identity matrix.

Similarly, define a strain tensor associated with the frame $Oxyz$ by

$$[\varepsilon] = \begin{bmatrix} \varepsilon_x & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & \varepsilon_y & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & \varepsilon_z \end{bmatrix} = \begin{bmatrix} \varepsilon_x & \frac{1}{2}\gamma_{xy} & \frac{1}{2}\gamma_{xz} \\ \frac{1}{2}\gamma_{xy} & \varepsilon_y & \frac{1}{2}\gamma_{yz} \\ \frac{1}{2}\gamma_{xz} & \frac{1}{2}\gamma_{yz} & \varepsilon_z \end{bmatrix} \tag{15.9}$$

The transformation of the strain components from frame $Oxyz$ to frame $Ox'y'z'$ are obtained by

$$[\varepsilon'] = \begin{bmatrix} \varepsilon_{x'} & \varepsilon_{x'y'} & \varepsilon_{x'z'} \\ \varepsilon_{x'y'} & \varepsilon_{y'} & \varepsilon_{y'z'} \\ \varepsilon_{x'z'} & \varepsilon_{y'z'} & \varepsilon_{z'} \end{bmatrix} = [T]^T [\varepsilon][T] \tag{15.10}$$

where the matrix of direction cosines is given by Eq. (15.8).

Refer to Chapter 5 for transformation of stress and strain in two dimensions, and for Mohr's circle for stress and strain.

Principal Stresses For any state of stress, there is a frame $Ox'y'z'$ such that the shear stresses vanish; that is, $\tau_{x'y'} = 0, \tau_{x'z'} = 0, \tau_{y'z'} = 0$. In this case, the stress matrix is of the diagonal form

$$[\sigma'] = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \tag{15.11}$$

where the normal stresses σ_1, σ_2, and σ_3 are called *principal stresses*. The principal stresses are usually arranged in a descending order, $\sigma_1 \geq \sigma_2 \geq \sigma_3$, with σ_1 being the maximum principal stress and σ_3 the minimum principal stress.

Given a stress matrix $[\sigma]$ associated with frame $Oxyz$, the principal stresses and their directions (the principal axes of stress) are described by the following eigenvalue problem

$$[\sigma]\{d\} = \lambda\{d\} \tag{15.12}$$

where λ is an eigenvalue and $\{d\}$ is an eigenvector. The principal stresses are the roots of the characteristic equation

$$\det\{\lambda[I] - [\sigma]\} = \lambda^3 - I_1\lambda^2 + I_2\lambda + I_3 = 0 \tag{15.13}$$

namely, $\lambda = \sigma_1, \sigma_2, \sigma_3$, where $[I]$ is the identity matrix and

$$\begin{aligned} I_1 &= \sigma_x + \sigma_y + \sigma_z \\ I_2 &= \begin{vmatrix} \sigma_x & \tau_{xy} \\ \tau_{xy} & \sigma_y \end{vmatrix} + \begin{vmatrix} \sigma_x & \tau_{xz} \\ \tau_{xz} & \sigma_z \end{vmatrix} + \begin{vmatrix} \sigma_y & \tau_{yz} \\ \tau_{yz} & \sigma_z \end{vmatrix} \\ I_3 &= \det[\sigma] \end{aligned} \tag{15.14}$$

The I_1, I_2, and I_3 are called the *stress invariants* whose values are independent of the orientation of the coordinate axes chosen. The orientation of the principal axes is described by the

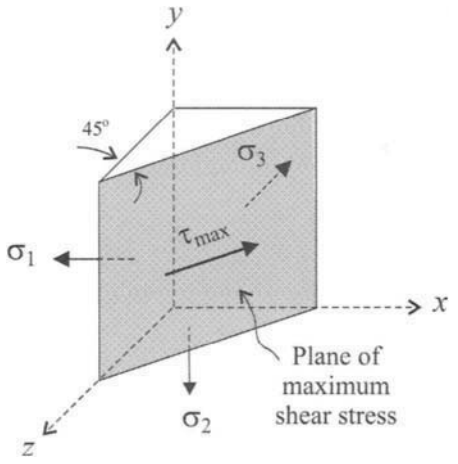


FIGURE 15.2.5 The plane of maximum shear stress.

matrix of direction cosines

$$[T] = \left[\{d_1\} \quad \{d_2\} \quad \{d_3\}\right]. \tag{15.15}$$

where $\{d_j\}$ is the eigenvector associated with the eigenvalue $\lambda = \sigma_j$ and satisfies the condition

$$\{d_j\}^T \{d_j\} = 1, \quad j = 1, 2, 3 \tag{15.16}$$

Maximum Shear Stress Let the axes of frame $Oxyz$ be the principal axes of stress; see Fig. 15.2.5, where $\sigma_1 \geq \sigma_2 \geq \sigma_3$. The maximum shear stress is

$$\tau_{\max} = \frac{1}{2}(\sigma_1 - \sigma_3) \tag{15.17}$$

which acts on a plane with the normal that is perpendicular to the y axis and makes 45° (or −45°) with the x axis. In other words, the plane of maximum shear stress bisects the planes of maximum and minimum principal stresses.

Principal Strains For any state of strain, there exists a frame $Ox'y'z'$ under which the shear strains vanish; namely

$$[\varepsilon'] = \begin{bmatrix} \varepsilon_1 & 0 & 0 \\ 0 & \varepsilon_2 & 0 \\ 0 & 0 & \varepsilon_3 \end{bmatrix} \tag{15.18}$$

where the normal strains ε_1, ε_2, and ε_3 are known as *principal strains*. Given a strain matrix $[\varepsilon]$ associated with frame $Oxyz$, the principal strains are the roots of the characteristic equation

$$\det\{\lambda [I] - [\varepsilon]\} = \lambda^3 - J_1\lambda^2 + J_2\lambda + J_3 = 0 \tag{15.19}$$

with

$$J_1 = \varepsilon_x + \varepsilon_y + \varepsilon_z$$

$$J_2 = \begin{vmatrix} \varepsilon_x & \varepsilon_{xy} \\ \varepsilon_{xy} & \varepsilon_y \end{vmatrix} + \begin{vmatrix} \varepsilon_x & \varepsilon_{xz} \\ \varepsilon_{xz} & \varepsilon_z \end{vmatrix} + \begin{vmatrix} \varepsilon_y & \varepsilon_{yz} \\ \varepsilon_{yz} & \varepsilon_z \end{vmatrix} \tag{15.20}$$

$$J_3 = \det[\varepsilon]$$

The J_1, J_2, and J_3 are the strain invariants whose values remain the same regardless of the orientation of the coordinate axes chosen. The principal axes of strain are determined by the matrix of direction cosines expressed by Eqs. (15.15) and (15.16), where $\{d_j\}$ are the eigenvectors of the eigenequation

$$[\varepsilon]\{d\} = \lambda\{d\} \tag{15.21}$$

The MATLAB Toolbox of this chapter has functions `pristress3` and `pristrain3` for determination of principal stresses and strains; see Windows 2.1 and 2.2.

Window 2.1. Function `pristress3`

MATLAB Function: **`pristress3`**

Purpose:
To compute principal stresses and principal axes of stress.

Synopsis:

```
pristress(s)
[S_prin, Ax_prin, S_inv] = pristress(s)
```

Description:
`pristress(s)` computes the principal stresses for a state of stress given by the vector `s` $= [\sigma_x\ \sigma_y\ \sigma_z\ \tau_{xy}\ \tau_{xz}\ \tau_{yz}]$, gives the maximum shear stress by Eq. (15.17), displays in a figure the orientation of the principal axes of stress, and shows the stress invariants I_1, I_2, and I_3.

`[S_prin, Ax_prin, S_inv] = pristress(s)` returns the principal stresses in a vector `S_prin`, the direction cosines for the principal axes in a matrix `Ax_Prin`, and the stress invariants in a vector `S_inv`.

Window 2.2. Function `pristrain3`

MATLAB Function: **`pristrain3`**

Purpose:
To compute principal strains and principal axes of strain.

Synopsis:

```
pristrain(s)
[S_prin, Ax_prin, S_inv] = pristrain(s)
```

Window 2.2. Function `pristrain3` (*Continued*)

Description:
`pristrain(s)` computes the principal strains for a state of strain given by the vector `s` = $[\varepsilon_x\ \varepsilon_y\ \varepsilon_z\ \gamma_{xy}\ \gamma_{xz}\ \gamma_{yz}]$, displays in the figure the orientation of the principal axes of strain, and shows the strain invariants J_1, J_2, and J_3.

`[S_prin, Ax_prin, S_inv] = pristrain(s)` returns the principal strains in a vector `S_prin`, the direction cosines for the principal axes in a matrix `Ax_Prin`, and the stress invariants in a vector `S_inv`.

EXAMPLE 2.1

A state of stress in the frame *Oxyz* is given by the matrix

$$[\sigma] = \begin{bmatrix} -18 & -5 & 7 \\ -5 & 6 & 12 \\ 7 & 12 & -9 \end{bmatrix} \text{MPa}$$

where 1 MPa = 10^6 Pa. The principal stresses and their orientation, and the stress invariants are computed by the following command typed in the MATLAB command window

```
>> pristress3([-18 6 -9 -5 7 12])
```

which yields

```
Principal Stresses:
 S1 = 12.6851
 S2 = -8.2187
 S3 = -25.4665
Maximum Shear Stress:
 Tmax = 38.1516
Stress Invariants:
 I1 = -21
 I2 = -218
 I3 = 2655
Principal Axes x', y', and z':
 (direction cosines between xyz and x'y'z')
          x'          y'          z'
    ------------------------------
x    0.034837    -0.6559     0.75404
y   -0.87916      0.33869    0.33522
z   -0.47526     -0.6746    -0.56484
```

and the plot of the principal axes in Fig. 15.2.6. The computed stresses are in MPa.

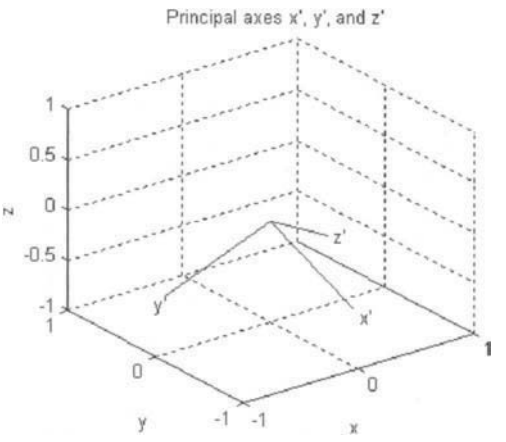


FIGURE 15.2.6

15.2.2 Basic Equations of Elasticity

The formulation and solution of an elasticity problem requires the following basic equations:

- Equations of equilibrium
- Strain-displacement relations (geometric relations)
- Conditions of compatibility
- Stress-strain relations (constitutive law)
- Boundary conditions

Equations of Equilibrium For a body in static equilibrium, the stress components at every point should satisfy the differential equations

$$\begin{aligned} \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} + F_x &= 0 \\ \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yz}}{\partial z} + F_y &= 0 \\ \frac{\partial \sigma_z}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + F_z &= 0 \end{aligned} \tag{15.22}$$

where F_x, F_y, and F_z are the x, y, and z components of the body forces (forces per unit volume) applied to the body.

Strain-Displacement Relations For a body in small deformation, its strains are expressed by

$$\begin{gathered} \varepsilon_x = \frac{\partial u}{\partial x}, \quad \varepsilon_y = \frac{\partial v}{\partial y}, \quad \varepsilon_z = \frac{\partial w}{\partial z} \\ \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}, \quad \gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}, \quad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \end{gathered} \tag{15.23}$$

where u, v, and w are the components of the displacement of the body, as given in Eq. (15.4).

Stress-Strain Relations For a body of isotropic linearly elastic material, its stress and strain components satisfy (Reference 4)

$$\begin{aligned}
\varepsilon_x &= \frac{1}{E}\left[\sigma_x - \mu(\sigma_y + \sigma_z)\right], \quad \gamma_{xy} = \frac{1}{G}\tau_{xy} \\
\varepsilon_y &= \frac{1}{E}\left[\sigma_y - \mu(\sigma_x + \sigma_z)\right], \quad \gamma_{yz} = \frac{1}{G}\tau_{yz} \\
\varepsilon_z &= \frac{1}{E}\left[\sigma_z - \mu(\sigma_x + \sigma_y)\right], \quad \gamma_{xz} = \frac{1}{G}\tau_{xz}
\end{aligned} \tag{15.24}$$

where E is Young's modulus, μ is Poisson's ratio, and G is shear modulus or modulus of rigidity. The shear modulus is related to Young's modulus and Poisson's ratio by

$$G = \frac{E}{2(1+\mu)}. \tag{15.25}$$

The units of E and G are the same as stresses; μ is a dimensionless quantity. Equations (15.24) are known as the *generalized Hooke's law*. Appendix E gives the values of elastic coefficients E, G, and μ for some engineering materials.

Equations (15.24) can be inverted to express stresses in terms of strains

$$\begin{aligned}
\sigma_x &= \frac{E}{(1+\mu)(1-2\mu)}\left[(1-\mu)\varepsilon_x + \mu(\varepsilon_y + \varepsilon_z)\right], \quad \tau_{xy} = G\gamma_{xy} \\
\sigma_y &= \frac{E}{(1+\mu)(1-2\mu)}\left[(1-\mu)\varepsilon_y + \mu(\varepsilon_x + \varepsilon_z)\right], \quad \tau_{yz} = G\gamma_{yz} \\
\sigma_z &= \frac{E}{(1+\mu)(1-2\mu)}\left[(1-\mu)\varepsilon_z + \mu(\varepsilon_x + \varepsilon_y)\right], \quad \tau_{xz} = G\gamma_{xz}
\end{aligned} \tag{15.26}$$

Also, through introduction of the unit volume change or dilatation

$$e = \varepsilon_x + \varepsilon_y + \varepsilon_z \tag{15.27}$$

Eqs. (15.26) are reduced to

$$\begin{aligned}
\sigma_x &= \lambda e + 2G\varepsilon_x, \quad \tau_{xy} = G\gamma_{xy} \\
\sigma_y &= \lambda e + 2G\varepsilon_y, \quad \tau_{yz} = G\gamma_{yz} \\
\sigma_z &= \lambda e + 2G\varepsilon_z, \quad \tau_{xz} = G\gamma_{xz}
\end{aligned} \tag{15.28}$$

with

$$\lambda = \frac{\mu E}{(1+\mu)(1-2\mu)} \tag{15.29}$$

Parameters λ and G are called the *constants of Lamé*.

As an example, consider a cubic element subject to hydrostatic pressure p. The stress components are $\sigma_x = \sigma_y = \sigma_z = -p$ and $\tau_{xy} = \tau_{yz} = \tau_{xz} = 0$. By Eq. (15.28),

$$-p = Ke, \quad \text{with } K = \frac{E}{3(1-2\mu)} \tag{15.30}$$

where K is called the *bulk modulus of elasticity*.

The Toolbox of this chapter has functions `hks2e` and `hke2s` for computing strains and stress components based on the generalized Hooke's law; see Window 2.3.

Window 2.3. Functions hks2e and hke2s

MATLAB Functions: **hks2e, hke2s**

Purpose:
To compute strain or stress components in three dimensions by the generalized Hooke's law.

Synopsis:

```
hks2e(stress, E, Mu)
y = hks2e(stress, E, Mu)
hke2s(strain, E, Mu)
y = hke2s(strain, E, Mu)
```

Description:
`hks2e(stress, E, Mu)` computes strain components based on Eq. (15.24), where `stress` is a vector of given stress components, i.e., `stress` = $[\sigma_x \ \sigma_y \ \sigma_z \ \tau_{xy} \ \tau_{yz} \ \tau_{xz}]$; `E` is Young's modulus; and `Mu` is Poisson's ratio.

`y = hks2e(stress, E, Mu)` returns the computed strain components in a vector

$$\mathtt{y} = [\varepsilon_x \ \varepsilon_y \ \varepsilon_z \ \gamma_{xy} \ \gamma_{yz} \ \gamma_{xz}].$$

`hke2s(strain, E, Mu)` computes stress components based on Eq. (15.26), where `strains` is a vector of given components, i.e., `strains` = $[\varepsilon_x \ \varepsilon_y \ \varepsilon_z \ \gamma_{xy} \ \gamma_{yz} \ \gamma_{xz}]$; `E` is Young's modulus; and `Mu` is Poisson's ratio.

`y = hke2s(strain, E, Mu)` returns the computed stress components in a vector

$$\mathtt{y} = [\sigma_x \ \sigma_y \ \sigma_z \ \tau_{xy} \ \tau_{yz} \ \tau_{xz}].$$

EXAMPLE 2.2

Consider a material with $G = 80$ GPa and $E = 200$ GPa, in a state of stress

$$[\sigma] = \begin{bmatrix} 15 & -3 & 6 \\ -3 & 0 & 7 \\ 6 & 7 & 18 \end{bmatrix} \text{MPa}$$

where 1 GPa $= 10^9$ Pa. By the following MATLAB commands

```
>> E = 200*1e-3; G = 80*1e-3;
>> mu = E/G/2-1;
>> hks2e([15 0 18 -3 7 6], E, mu)
```

the strain components are computed as follows

```
Normal strain Ex = 52.5
Normal strain Ey = -41.25
Normal strain Ez = 71.25
Shear strain Lxy = -37.5
Shear strain Lyz = 87.5
Shear strain Lxz = 75
```

all of which are in μ (10^{-6}).

Conditions of Compatibility According to the strain-displacement relations (15.23), the six components of strain at a point are completely determined by the three displacement components u, v, and w. Therefore, the strain components cannot be arbitrary functions of x, y, and z. They must satisfy the following conditions of compatibility (Reference 4):

$$\begin{aligned}
&\frac{\partial^2\varepsilon_x}{\partial y^2}+\frac{\partial^2\varepsilon_y}{\partial x^2}=\frac{\partial^2\gamma_{xy}}{\partial x\partial y}, \quad 2\frac{\partial^2\varepsilon_x}{\partial y\partial z}=\frac{\partial}{\partial x}\left(-\frac{\partial\gamma_{yz}}{\partial x}+\frac{\partial\gamma_{xz}}{\partial y}+\frac{\partial\gamma_{xy}}{\partial z}\right)\\
&\frac{\partial^2\varepsilon_y}{\partial z^2}+\frac{\partial^2\varepsilon_z}{\partial y^2}=\frac{\partial^2\gamma_{yz}}{\partial y\partial z}, \quad 2\frac{\partial^2\varepsilon_y}{\partial z\partial x}=\frac{\partial}{\partial y}\left(\frac{\partial\gamma_{yz}}{\partial x}-\frac{\partial\gamma_{xz}}{\partial y}+\frac{\partial\gamma_{xy}}{\partial z}\right)\\
&\frac{\partial^2\varepsilon_z}{\partial x^2}+\frac{\partial^2\varepsilon_x}{\partial z^2}=\frac{\partial^2\gamma_{xz}}{\partial z\partial x}, \quad 2\frac{\partial^2\varepsilon_z}{\partial x\partial y}=\frac{\partial}{\partial z}\left(\frac{\partial\gamma_{yz}}{\partial x}+\frac{\partial\gamma_{xz}}{\partial y}-\frac{\partial\gamma_{xy}}{\partial z}\right)
\end{aligned} \tag{15.31}$$

Mathematically, the conditions (15.31) assert that the displacements are single-valued continuous functions that are associated with the strain components.

Boundary Conditions For a body B in three-dimensional space its boundary consists of two parts: the surface S_σ on which surface tractions (forces per unit area) are specified and the surface S_u on which displacements are specified; see Fig. 15.2.7. The equations specifying the surface tractions and displacements are called the *boundary conditions*, and they have the form

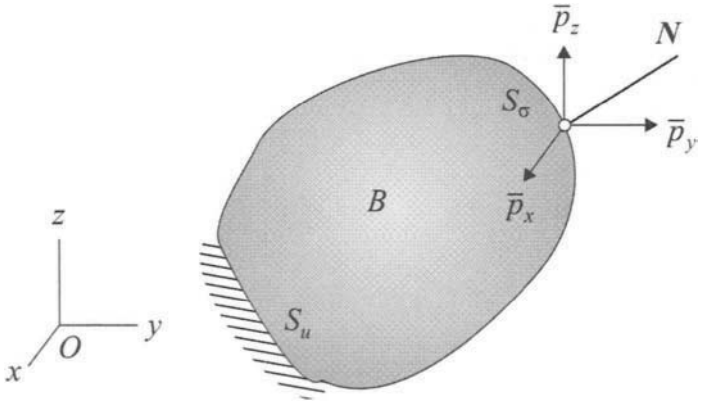


FIGURE 15.2.7 The boundary of a body.

Boundary conditions on S_σ

$$\begin{aligned} \sigma_x l + \tau_{xy} m + \tau_{xz} n &= \bar{p}_x \\ \tau_{xy} l + \sigma_y m + \tau_{yz} n &= \bar{p}_y \\ \tau_{xz} l + \tau_{yz} m + \sigma_z n &= \bar{p}_z \end{aligned} \tag{15.32a}$$

where $\bar{p}_x$, $\bar{p}_y$, and $\bar{p}_z$ are the x, y, and z components of the given surface tractions, and l, m, and n are the direction cosines of the surface normal N.

Boundary conditions on S_u

$$u = \bar{u}, \quad v = \bar{v}, \quad w = \bar{w} \tag{15.32b}$$

where $\bar{u}$, $\bar{v}$, and $\bar{w}$ are the prescribed displacements.

Saint-Venant's Principle In application of the elasticity theory to a practical problem, the description of exact boundary tractions often makes the problem too complicated to solve. In this case, the principle due to Saint-Venant may be applied. This principle may be stated as follows:

If a system of forces acting on a small portion of an elastic body is replaced by another statically equivalent system of forces acting on the same portion of the body, the significant changes in stress distribution only occur in the immediate neighborhood of the loading; the stresses in other parts of the body that are a distance from the region of the loading are essentially the same.

By Saint-Venant's principle, the conditions on boundary tractions are changed such that an elasticity solution can be obtained.

Formation of Elasticity Problem Consider an elastic body under body forces F_x, F_y, F_z, surface tractions $\bar{p}_x$, $\bar{p}_y$, $\bar{p}_z$ and/or displacements $\bar{u}$, $\bar{v}$, and $\bar{w}$ on the boundary. A fundamental problem in static analysis of the elastic body is to solve the equilibrium equations (15.22), subject to strain-displacement relations (15.23), stress-strain relations (15.28), and boundary conditions (15.32), for the displacement and stress components. The displacement and strain solutions must satisfy the conditions of compatibility (15.31).

Exact analytical solutions are only available for certain elastic continua of simple geometry. For many elasticity problems in engineering, solutions are often obtained by numerical methods such as the finite element method; see Section 15.4 for instance.

15.2.3 Conversion of Elasticity Constants

Five constants have been introduced to describe the mechanical properties of isotropic linearly elastic materials: E (Young's modulus), μ (Poisson's ratio), G (shear modulus), λ (constant of Lamé), and K (bulk modulus of elasticity). These elastic constants are related. For example, G, λ, and K are expressed in terms of E and μ in Eqs. (15.25), (15.29), and (15.30), namely,

$$G = \frac{E}{2(1+\mu)}, \quad \lambda = \frac{\mu E}{(1+\mu)(1-2\mu)}, \quad K = \frac{E}{3(1-2\mu)}.$$

Also, E, μ, and K can be expressed in terms of the constants of Lamé

$$E = \frac{G(3\lambda + 2G)}{\lambda + G}, \quad \mu = \frac{\lambda}{2(\lambda + G)}, \quad K = \lambda + \frac{2}{3}G.$$

In fact, any two of the five constants can be used to express the other three.

The Toolbox of this chapter provides a function `elaconst` for computing the values of the related elastic constants; see Window 2.4.

Window 2.4. Function `elaconst`

MATLAB Function: **`elaconst`**

Purpose:
To compute the values of related elasticity constants.

Synopsis:

```
elaconst(coeffs)
y = elaconst(coeffs)
```

Description:
`elaconst(coeffs)`, for any two elasticity constants selected among E, μ, G, λ, and K, computes the rest of the elastic constants. The `coeffs` is a 2-by-2 matrix describing the two selected constants and is of the form

```
coeffs = [Id_1 Par_1; Id_2 Par_2]
```

where `Id_k` is an integer from 1 to 5 identifying a constant as follows

`Id_k = 1` for E (Young's modulus)
`Id_k = 2` for μ (Poisson's ratio)
`Id_k = 3` for G (shear modulus)
`Id_k = 4` for λ (constant of Lamé)
`Id_k = 5` for K (bulk modulus of elasticity)

and `Par_k` specifies the value of the selected constant. For example

```
coeffs = [1 8000; 3 5000]
```

indicates that $E = 8{,}000$ and $G = 5{,}000$.

`y = elaconst(coeffs)` returns the elastic constants in a row vector `y`.

EXAMPLE 2.3

For a material with $G = 80$ GPa and $\lambda = 120$ GPa, the constants E, μ, and K are computed by

```
>> elaconst([3 80; 4 120])
Conversion of Elastic Constants
------------------------------
Given constants:
 Shear modulus G = 80
 Lame constant lambda = 120
Computed constants:
 Young's modulus E = 208
 Poisson's ratio mu = 0.3
 Bulk modulus of elasticity K = 173.3333
```

where E and K are in GPa

15.2.4 Strain Energy

Strain Energy Density Under external forces, an elastic body deforms. The net work done by these forces is stored in the body in the form of strain energy, which is given by the volume integral

$$U = \int_B U_d dB \tag{15.33}$$

where B is the region of the body and U_d is known as the *strain energy density function* (strain energy per unit volume). For linearly elastic materials,

$$U_d = \frac{1}{2}\left(\sigma_x\varepsilon_x + \sigma_y\varepsilon_y + \sigma_z\varepsilon_z + \tau_{xy}\gamma_{xy} + \tau_{yz}\gamma_{yz} + \tau_{xz}\gamma_{xz}\right) \tag{15.34}$$

For a body of isotropic linearly elastic material satisfying the generalized Hooke's law, its strain energy density function can be expressed by stress components

$$U_d = \frac{1}{2E}\left(\sigma_x^2 + \sigma_y^2 + \sigma_z^2\right) - \frac{1}{E}\left(\sigma_x\sigma_y + \sigma_y\sigma_z + \sigma_z\sigma_x\right) + \frac{1}{2G}\left(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{xz}^2\right) \tag{15.35}$$

or by strain components

$$U_d = \frac{1}{2}\left\{\lambda e^2 + 2G\left(\varepsilon_x^2 + \varepsilon_y^2 + \varepsilon_z^2\right) + G\left(\gamma_{xy}^2 + \gamma_{yz}^2 + \gamma_{xz}^2\right)\right\} \tag{15.36}$$

where $e = \varepsilon_x + \varepsilon_y + \varepsilon_z$ and λ and G are the constants of Lamé. It can be shown that

$$\sigma_x = \frac{\partial U_d}{\partial \varepsilon_x}, \quad \sigma_y = \frac{\partial U_d}{\partial \varepsilon_y}, \quad \sigma_z = \frac{\partial U_d}{\partial \varepsilon_z}, \quad \tau_{xy} = \frac{\partial U_d}{\partial \gamma_{xy}}, \quad \tau_{yz} = \frac{\partial U_d}{\partial \gamma_{yz}}, \quad \tau_{xz} = \frac{\partial U_d}{\partial \gamma_{xz}} \tag{15.37}$$

Strain Energy for One-Dimensional Continua For a one-dimensional elastic continuum such as the bar in longitudinal deformation, circular shaft in torsional deformation, and beam in bending, strain energy can be written as

$$U = \int_0^L u_d \, dx \tag{15.38}$$

where L is the length of the continuum and u_d is the strain energy per unit length. Table 15.2.1 shows the strain energy for three types of one-dimensional elastic members. Refer to Chapters 2 and 3 for more information on these members.

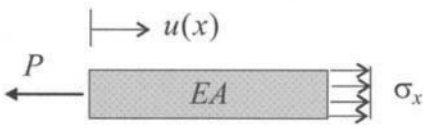
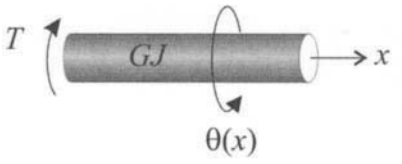
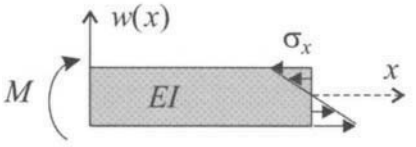
15.2.5 Principal of Minimum Potential Energy

The following discussion makes use of the knowledge in the calculus of variations.

Virtual Displacement of a Particle Consider a particle P that is in equilibrium under n forces $\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_n$. Let $\delta\boldsymbol{r}$ be an arbitrary displacement of the particle, which may not actually take place. By the principal of virtual displacement, the total work done by these forces during any virtual displacement is zero, namely

$$\delta W = \boldsymbol{f}_1 \cdot \delta\boldsymbol{r} + \boldsymbol{f}_2 \cdot \delta\boldsymbol{r} + \cdots + \boldsymbol{f}_n \cdot \delta\boldsymbol{r} = 0 \tag{15.39}$$

TABLE 15.2.1 Strain Energy of Elastic Members

Member	*Stress and Strain Energy*
Bar in longitudinal deformation  u – longitudinal displacement P – axial force	Normal stress $\sigma_x = \frac{P}{A} = E\frac{du}{dx}$ $U = \frac{1}{2}\int_B \frac{\sigma_x^2}{E} dB = \frac{1}{2}\int_0^L EA\left(\frac{du}{dx}\right)^2 dx$ $u_d = \frac{1}{2}EA\left(\frac{du}{dx}\right)^2$
Circular shaft in torsion  θ – rotation angle T – torque	Shear stress $\tau = \frac{T}{J}r = Gr\frac{d\theta}{dx}$, $J = \int_A r^2 dA$ $U = \frac{1}{2}\int_B \frac{\tau^2}{G} dB = \frac{1}{2}\int_0^L GJ\left(\frac{d\theta}{dx}\right)^2 dx$ $u_d = \frac{1}{2}GJ\left(\frac{d\theta}{dx}\right)^2$
Beam in bending  w – transverse displacement M – bending moment	Normal stress $\sigma_x = -\frac{My}{I} = -Ey\frac{d^2w}{dx^2}$, $I = \int_A y^2 dA$ $U = \frac{1}{2}\int_B \frac{\sigma_x^2}{E} dB = \frac{1}{2}\int_0^L EI\left(\frac{d^2w}{dx^2}\right)^2 dx$ $u_d = \frac{1}{2}EI\left(\frac{d^2w}{dx^2}\right)^2$

which yields the equilibrium equation

$$\boldsymbol{f}_1 + \boldsymbol{f}_2 + \cdots + \boldsymbol{f}_n = 0 \tag{15.40}$$

Variation of Potential Energy of an Elastic Body Extend the concept of virtual displacement to a deformed elastic body with actual displacement components u, v, and w. Let δu, δv, and δw be the components of an arbitrary and virtual displacement of the body, which satisfy the following conditions:

(C1) δu, δv, and δw are continuous functions of x, y, and z; and
(C2) δu, δv, and δw are zero on the part S_u of the boundary where displacements are prescribed, as shown in Eq. (15.32b).

The above-described virtual displacement need not actually take place and does not have to be infinitesimal. Displacements satisfying (C1) and (C2) shall be called *admissible displacements*.

The δ, in the calculus of variations, is the operator of variation, for which the rules of operation follow those for the differential operator d. For instance, if H is a function of u, v, and w, which is often called a *functional*, its variation due to δu, δv, and δw is expressed by

$$\delta H = \frac{\partial H}{\partial u}\delta u + \frac{\partial H}{\partial v}\delta v + \frac{\partial H}{\partial w}\delta w \tag{15.41}$$

Furthermore, the changes of the strain components due to the virtual displacement are given by

$$\delta\varepsilon_x = \delta\left(\frac{\partial u}{\partial x}\right) = \frac{\partial \delta u}{\partial x}, \ldots, \quad \delta\gamma_x = \delta\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) = \frac{\partial \delta u}{\partial y} + \frac{\partial \delta v}{\partial x}, \ldots \tag{15.42}$$

where δ and $\partial/\partial x$ are exchangeable according to the calculus of variations.

Now, define a potential energy for the elastic body

$$\Pi = U - W = U - \int_B \left(F_x u + F_y v + F_z w\right) dB - \int_{S_\sigma} \left(\bar{p}_x u + \bar{p}_y v + \bar{p}_z w\right) dS \tag{15.43}$$

where U is the strain energy of the body as given by Eq. (15.33), and $-W$ is the potential of the actual body forces F_x, F_y, F_z, and the prescribed surface tractions $\bar{p}_x$, $\bar{p}_y$, $\bar{p}_z$ on the part S_σ of the boundary as shown in Eq. (15.32a). Note that Π is a functional with u, v, and w as function arguments. Assume that the external forces do not change during the virtual displacements δu, δv, and δw. The variation of Π due to the virtual displacement is

$$\delta\Pi = \delta U - \delta W \tag{15.44}$$

where

$$\delta U = \int_B \left(\sigma_x \delta\varepsilon_x + \sigma_y \delta\varepsilon_y + \sigma_z \delta\varepsilon_z + \tau_{xy}\delta\gamma_{xy} + \tau_{yz}\delta\gamma_{yz} + \tau_{xz}\delta\gamma_{xz}\right) dB \tag{15.45}$$

is the variation of the strain energy and

$$\delta W = \int_B \left(F_x \delta u + F_y \delta v + F_z \delta w\right) dB + \int_{S_\sigma} \left(\bar{p}_x \delta u + \bar{p}_y \delta v + \bar{p}_z \delta w\right) dS \tag{15.46}$$

is the virtual work done by the external forces.

Principal of Minimum Potential Energy The Principal of Minimum Potential Energy (PMPE) is stated as follows:

Of all admissible displacements, the actual ones, which satisfy the equilibrium equations (15.22), render the potential energy stationary, that is,

$$\delta\Pi = \delta U - \delta W = 0 \tag{15.47}$$

Furthermore, for stable equilibrium, the actual displacements make the potential energy minimum.

For a body of isotropic linearly elastic material, the second variation of its potential energy, by Eq. (15.36), is

$$\delta^2\Pi = \int_B \left[\lambda\delta e^2 + 2G\left(\delta\varepsilon_x^2 + \delta\varepsilon_y^2 + \delta\varepsilon_z^2\right) + G\left(\delta\gamma_{xy}^2 + \delta\gamma_{yz}^2 + \delta\gamma_{xz}^2\right)\right] dB \tag{15.48}$$

which is positive definite for any admissible displacement. This implies that for an isotropic linearly elastic body, its potential energy is a minimum at equilibrium.

The PMPE has two important applications. First, the PMPE lays a foundation for numerical solution methods for elasticity problems such as the Rayleigh-Ritz method and the finite element method; see Section 15.4 for instance. Second, the PMPE can be used to derive equilibrium equations and boundary conditions for deformable bodies, as shown in the following example.

EXAMPLE 2.4

In Fig. 15.2.8, a nonuniform bar with fixed and spring-supported ends undergoes longitudinal deformation under an axial load $p(x)$ and a lumped force $\bar{Q}$ at the right end. The equilibrium equation and boundary conditions for the bar can be derived by the PMPE. To this end, write the potential energy of the bar as follows

$$\Pi = \frac{1}{2}\int_0^L EA\left(\frac{du}{dx}\right)^2 dx + \frac{1}{2}ku^2(L) - \int_0^L pu\,dx - \bar{Q}u(L)$$

where the first term is the strain energy of the bar (see Table 15.2.1); the second term is the potential energy of the end spring; and the last two terms are the potential energy of the external forces. The variation of the potential energy of the bar gives

$$\delta\Pi = \int_0^L EA\frac{du}{dx}\frac{d\delta u}{dx}dx + ku(L)\delta u(L) - \int_0^L p\,\delta u\,dx - \bar{Q}\delta u(L)$$

$$= -\int_0^L \left[\frac{d}{dx}\left(EA\frac{du}{dx}\right) + p\right]\delta u dx - EA\frac{du}{dx}\delta u\Big|_{x=0} + \left[EA\frac{du}{dx} - \bar{Q}\right]_{x=L}\delta u(L)$$

where integral by part has been applied. Because δu is an arbitrary virtual displacement, it should satisfy the boundary condition at the left end ($\delta u|_{x=0} = 0$). Thus, use of Eq. (15.47) yields the equilibrium equation

$$\frac{d}{dx}\left(EA\frac{du}{dx}\right) + p = 0, \quad 0 < x < L$$

and the boundary condition of the bar at the right end

$$EA\frac{du}{dx}\Big|_{x=L} = \bar{Q}$$

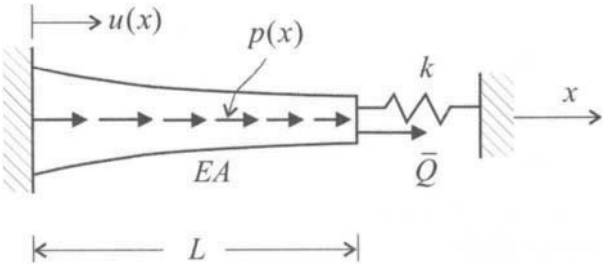


FIGURE 15.2.8

15.3.1 Plane Stress and Plane Strain

Depending on the geometric configuration of the elastic body in consideration, a two-dimensional elasticity problem usually falls into one of the two categories: plane stress and plane strain.

Plane Stress Problem The elastic body in consideration is a thin elastic plate subject to in-plane external forces that are uniformly distributed over the thickness; see Fig. 15.3.1. The stress components related to the z-axis are assumed zero, namely,

$$\sigma_z = 0, \quad \tau_{xz} = 0, \quad \tau_{yz} = 0 \tag{15.49}$$

Under this assumption, the stress-strain relations for the in-plane components, by Eq. (15.24), are

$$\varepsilon_x = \frac{1}{E}(\sigma_x - \mu\sigma_y), \quad \varepsilon_y = \frac{1}{E}(\sigma_y - \mu\sigma_x), \quad \gamma_{xy} = \frac{1}{G}\tau_{xy} \tag{15.50}$$

and the strain components related to the z-axis become

$$\varepsilon_z = -\frac{\mu}{E}(\sigma_x + \sigma_y) = -\frac{\mu}{1-\mu}(\varepsilon_x + \varepsilon_y), \quad \gamma_{xz} = 0, \quad \gamma_{yz} = 0 \tag{15.51}$$

Plane Strain Problem A problem of plane strain is concerned with a long prismatic or cylindrical body as shown in Fig. 15.3.2. The body is subject to external loads that are perpendicular to the longitudinal (z) axis, and do not change along the length. It is assumed that frictionless constraints are imposed at the two ends of the body, which permit x, y deformation, but do not allow the displacement in the z direction. Under the circumstances, the z displacement component is zero at every point of the body, i.e., $w = 0$; and so are the strain components related to the z-axis

$$\varepsilon_z = 0, \quad \gamma_{xz} = 0, \quad \gamma_{yz} = 0 \tag{15.52}$$

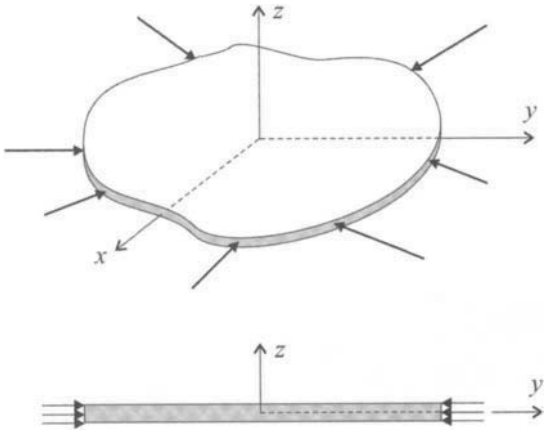


FIGURE 15.3.1 A body in plane stress.

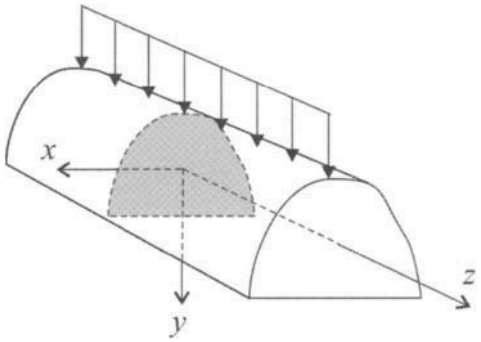


FIGURE 15.3.2 A body in plane strain.

The other strain components $\varepsilon_x, \varepsilon_y, \gamma_{xy}$ are only dependent upon x and y. So, a state of plane strain of the body can be represented by any cross-section (xy plane), as shown in Fig. 15.3.2.

In the case of plane strain, the stress-strain relations for the components in the xy plane are

$$\varepsilon_x = \frac{1-\mu^2}{E}\left(\sigma_x - \frac{\mu}{1-\mu}\sigma_y\right), \quad \varepsilon_y = \frac{1-\mu^2}{E}\left(\sigma_y - \frac{\mu}{1-\mu}\sigma_x\right), \quad \gamma_{xy} = \frac{1}{G}\tau_{xy} \tag{15.53}$$

which can be obtained by substituting Eqs. (15.52) into Eqs. (15.26). The stress components related to the longitudinal direction are

$$\sigma_z = \lambda\left(\varepsilon_x + \varepsilon_y\right) = \mu\left(\sigma_x + \sigma_y\right), \quad \tau_{xz} = 0, \quad \tau_{yz} = 0 \tag{15.54}$$

Stress and Strain Transformation Refer to Chapter 5 for stress/strain transformation in two dimensions, and for determination of principal stresses and maximum shear stress.

15.3.2 Governing Equations

This section lists the basic equations and relations for formulation and solution of elasticity problems in plane stress and plane strain.

Equilibrium Equations

$$\begin{aligned} \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + F_x = 0 \\ \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + F_y = 0 \end{aligned} \tag{15.55}$$

where F_x and F_y are the x, y components of body forces.

Strain-Displacement Relations

$$\varepsilon_x = \frac{\partial u}{\partial x}, \quad \varepsilon_y = \frac{\partial v}{\partial y}, \quad \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \tag{15.56}$$

Stress-Strain Relations According to Eqs. (15.50) and (15.53), the stress-strain relations can be written in a unified matrix form

$$\begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix} = \frac{1}{\tilde{E}} \begin{bmatrix} 1 & -\tilde{\mu} & 0 \\ -\tilde{\mu} & 1 & 0 \\ 0 & 0 & 2(1+\tilde{\mu}) \end{bmatrix} \begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} \tag{15.57a}$$

or

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} = \frac{\tilde{E}}{1-\tilde{\mu}^2} \begin{bmatrix} 1 & \tilde{\mu} & 0 \\ \tilde{\mu} & 1 & 0 \\ 0 & 0 & (1+\tilde{\mu})/2 \end{bmatrix} \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix} \tag{15.57b}$$

where $\tilde{E} = E, \tilde{\mu} = \mu$ for plane stress, and $\tilde{E} = \frac{E}{1-\mu^2}, \tilde{\mu} = \frac{\mu}{1-\mu}$ for plane strain. It can be shown that

$$\frac{\tilde{E}}{2(1+\tilde{\mu})} = \frac{E}{2(1+\mu)} = G \tag{15.58}$$

Condition of Compatibility

$$\frac{\partial^2 \varepsilon_x}{\partial y^2} + \frac{\partial^2 \varepsilon_y}{\partial x^2} = \frac{\partial^2 \gamma_{xy}}{\partial x \partial y} \tag{15.59}$$

Substitution of Eq. (15.57a) into Eq. (15.59) and use of Eq. (15.55) yields the compatibility condition in terms of stress components

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)(\sigma_x + \sigma_y) = -(1+\tilde{\mu})\left(\frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y}\right) \tag{15.60}$$

Boundary Conditions The boundary conditions are

$$\begin{aligned} \sigma_x l + \tau_{xy} m &= \bar{p}_x \\ \tau_{xy} l + \sigma_y m &= \bar{p}_y \end{aligned} \tag{15.61a}$$

for prescribed tractions and

$$u = \bar{u}, \quad v = \bar{v} \tag{15.61b}$$

for prescribed displacements. Here l and m are the direction cosines between the normal $\boldsymbol{n}$ of the boundary and the x- and y- axes; see Fig. 15.3.3.

15.3.3 Solution by Stress Function

Airy Stress Function In the case of zero body forces, the equilibrium equations (15.55) and compatibility condition (15.60) become

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} = 0, \quad \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} = 0 \tag{15.62}$$

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)(\sigma_x + \sigma_y) = 0 \tag{15.63}$$

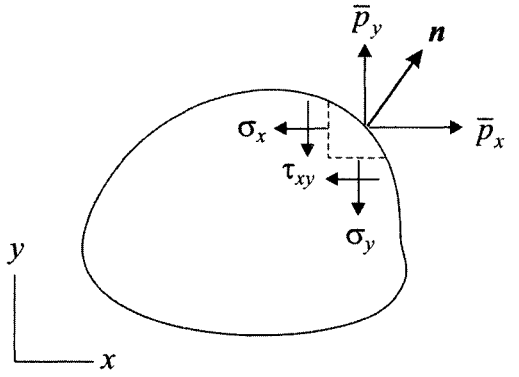


FIGURE 15.3.3 The boundary of a body in two dimensions.

Introduce the Airy stress function $\Phi(x, y)$ that is related to the stress components by

$$\sigma_x = \frac{\partial^2 \Phi}{\partial y^2}, \quad \sigma_y = \frac{\partial^2 \Phi}{\partial x^2}, \quad \tau_{xy} = -\frac{\partial^2 \Phi}{\partial x \partial y} \tag{15.64}$$

which automatically satisfy the equilibrium equations (15.62). Substitute Eq. (15.64) into the compatibility condition (15.63) to obtain the biharmonic equation

$$\nabla^4 \Phi = \frac{\partial^4 \Phi}{\partial x^4} + 2\frac{\partial^4 \Phi}{\partial x^2 \partial y^2} + \frac{\partial^4 \Phi}{\partial y^4} = 0 \tag{15.65}$$

Thus, for a two-dimensional elasticity problem with absent body forces, its solution is given by the stress function $\Phi(x, y)$ that is determined from Eq. (15.65) under appropriate boundary conditions.

Polynomial Solutions Certain elasticity problems can be solved by the semi-inverse method, in which part of the solution is first assumed or guessed and the rest is determined by the boundary conditions. The following polynomials are some candidates of guessed solutions:

Polynomial of the 2nd degree: $\Phi_2 = a_2 x^2 + b_2 xy + c_2 y^2$

Polynomial of the 3rd degree: $\Phi_3 = a_3 x^3 + b_3 x^2 y + c_3 x y^2 + d_3 y^3$

Polynomial of the 4th degree: $\Phi_4 = a_4 x^4 + b_4 x^3 y + c_4 x^2 y^2 + d_4 x y^3 + e_4 y^4$

.....

The unknown coefficients of these polynomials are related by Eq. (15.65), and are determined by the boundary conditions (References 4 and 5).

EXAMPLE 3.1

In Fig. 15.3.4, a cantilever of narrow rectangular cross-section is subject to a force P at its free end. The boundary conditions of the body are

$$\text{at } x = 0: \quad \sigma_x = 0, \quad -b \int_{-h/2}^{h/2} \tau_{xy}\, dy = P$$
$$\text{at } x = L: \quad u = 0, \quad v = 0$$
$$\text{at } y = \pm h/2: \quad \sigma_y = 0, \quad \tau_{xy} = 0$$

where Saint-Venant's principle has been applied to treat the end load P. Consider a stress function of the form (Reference 4)

$$\Phi = b_2 xy + d_4 xy^3$$

which automatically satisfies the biharmonic equation. The stress components, by Eq. (15.64), are

$$\sigma_x = 6d_4 xy, \quad \sigma_y = 0, \quad \tau_{xy} = -b_2 - 3d_4 y^2$$

which satisfy the boundary conditions $\sigma_x|_{x=0} = 0$ and $\sigma_y|_{y=\pm h/2} = 0$. The other boundary conditions

$$\tau_{xy}\big|_{y=\pm h/2} = -b_2 - 3d_4 \frac{h^2}{4} = 0 \quad \text{and} \quad -b \int_{-h/2}^{h/2} \tau_{xy}\, dy = b\left(b_2 h + \frac{1}{4} d_4 h^3\right) = P$$

lead to

$$b_2 = \frac{3P}{2bh}, \quad d_4 = -\frac{2P}{bh^3}.$$

Thus, the stress solution of the cantilever is

$$\sigma_x = -\frac{Pxy}{I}, \quad \sigma_y = 0, \quad \tau_{xy} = -\frac{P}{2I}\left(\frac{h^2}{4} - y^2\right)$$

where $I = bh^3/12$ is the moment of inertia of the cross-section about the neutral (z) axis. Also, through integration of Eq. (15.56), the displacements of the cantilever can be obtained.

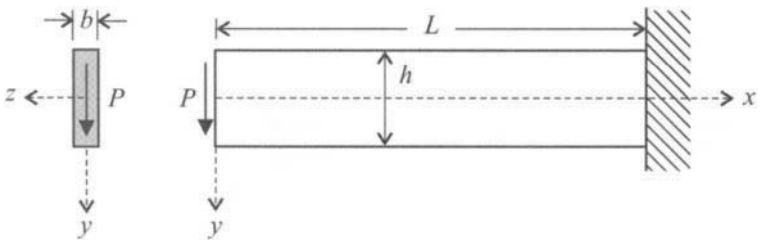


FIGURE 15.3.4

Exact analytical solutions by inverse or semi-inverse methods are limited to relatively simple problems. For solutions of general elasticity problems, numerical solution methods such as finite element methods are utilized; see Section 15.4.

15.3.4 Thermal Stresses

Besides external forces, nonuniform temperature can also cause stresses in an elastic body, which are called *thermal stresses*. Assume that T_0 is a uniform temperature at which the elastic body is unstrained. Let $T(x, y)$ be a change in temperature of the body above T_0. The strain induced by the temperature change is expressed by

$$\begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix} = \frac{1}{\tilde{E}} \begin{bmatrix} 1 & -\tilde{\mu} & 0 \\ -\tilde{\mu} & 1 & 0 \\ 0 & 0 & 2(1+\tilde{\mu}) \end{bmatrix} \begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} + \alpha T \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \tag{15.66}$$

where α is the coefficient of linear thermal expansion, which is defined as the change in length per unit length per degree rise in temperature; $\tilde{E} = E, \tilde{\mu} = \mu$ for plane stress and $\tilde{E} = \frac{E}{1-\mu^2}, \tilde{\mu} = \frac{\mu}{1-\mu}$ for plane strain. Here, for isotropic materials, shear strain is not affected by temperature change. Inverse Eq. (15.66) to give the stresses in terms of strain components

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} = \frac{\tilde{E}}{1-\tilde{\mu}^2} \begin{bmatrix} 1 & \tilde{\mu} & 0 \\ \tilde{\mu} & 1 & 0 \\ 0 & 0 & (1+\tilde{\mu})/2 \end{bmatrix} \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix} - \frac{\tilde{E}\alpha T}{1-\tilde{\mu}} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \tag{15.67}$$

Substituting Eq. (15.66) into Eq. (15.59) and neglecting body forces yields

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)(\sigma_x + \sigma_y + \alpha E T) = 0 \tag{15.68}$$

Introduce the stress function defined by Eq. (15.64), to have the compatibility equation

$$\nabla^4 \Phi + \alpha E \nabla^2 T = 0 \tag{15.69}$$

where the operator $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. Thus, thermal stresses can be determined from the solution of Eq. (15.69) subject to appropriate boundary conditions.

EXAMPLE 3.2

Consider a rectangular thin plate of length L, height h, and thickness t, which is subject to an arbitrary temperature $T(y)$ along the y-direction (Fig. 15.3.5). The plate is constrained by rigid walls at its two ends. For this problem of plane stress, the temperature-induced stress is determined as follows.

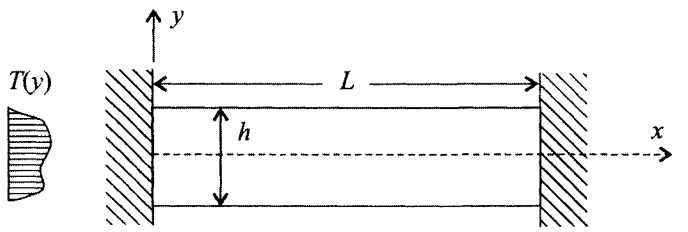


FIGURE 15.3.5

Assume that $\sigma_y = 0$, $\tau_{xy} = 0$, and $\varepsilon_x = 0$, and that σ_x is a function of y only. The compatibility condition (15.68) becomes

$$\frac{d^2}{dy^2}(\sigma_x + \alpha E T) = 0.$$

Integration of the above equation leads to

$$\sigma_x = -\alpha E T + a_1 y + a_2$$

With $\varepsilon_x = 0$ and $\sigma_y = 0$, Eq. (15.66) gives

$$\varepsilon_x = \frac{\sigma_x}{E} + \alpha T = \frac{a_1 y + a_2}{E} = 0, \quad \text{for} -\frac{h}{2} \le y \le \frac{h}{2}$$

which indicates that $a_1 = a_2 = 0$. It follows that

$$\sigma_x = -E\alpha T, \quad \varepsilon_y = (1+\mu)\alpha T, \quad \varepsilon_x = \gamma_{xy} = 0.$$

15.3.5 Elasticity Problems in Polar Coordinates

Stress and Strain in Polar Coordinates In elasticity problems, polar coordinates are convenient in describing stress and displacement of bodies of circular or annular shapes. The stress at a point is defined in the radial (r) direction and circumferential (θ) direction; see Fig. 15.3.6, where σ_r and σ_θ are normal stresses, $\tau_{r\theta}$ is shear stress, and F_r and F_θ are the components of body forces. The strain components ε_r, ε_θ, and $\gamma_{r\theta}$ are defined in a similar manner. The stress and strain in the polar coordinates (r, θ) are related to those in the rectangular coordinates (x, y) by

$$\begin{bmatrix} \sigma_r & \tau_{r\theta} \\ \tau_{r\theta} & \sigma_\theta \end{bmatrix} = [T]^T \begin{bmatrix} \sigma_x & \tau_{xy} \\ \tau_{xy} & \sigma_y \end{bmatrix} [T] \tag{15.70a}$$

$$\begin{bmatrix} \varepsilon_r & \gamma_{r\theta} \\ \gamma_{r\theta} & \varepsilon_\theta \end{bmatrix} = [T]^T \begin{bmatrix} \varepsilon_x & \gamma_{xy} \\ \gamma_{xy} & \varepsilon_y \end{bmatrix} [T] \tag{15.70b}$$

where

$$[T] = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \text{with } [T]^{-1} = [T]^T. \tag{15.71}$$

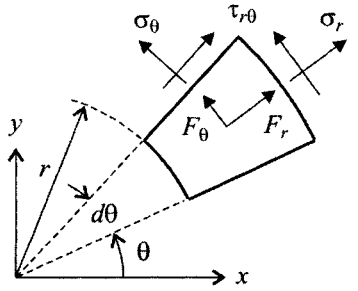


FIGURE 15.3.6 Stress components in polar coordinates.

Basic Equations A two-dimensional elasticity problem in the polar coordinates is described by the following equations and relations.

Equilibrium Equations

$$\begin{aligned} &\frac{\partial \sigma_r}{\partial r} + \frac{1}{r}\frac{\partial \tau_{r\theta}}{\partial \theta} + \frac{\sigma_r - \sigma_\theta}{r} + F_r = 0 \\ &\frac{1}{r}\frac{\partial \sigma_\theta}{\partial \theta} + \frac{\partial \tau_{r\theta}}{\partial r} + \frac{2\tau_{r\theta}}{r} + F_\theta = 0 \end{aligned} \tag{15.72}$$

Strain-Displacement Relations

$$\varepsilon_r = \frac{\partial u}{\partial r}, \quad \varepsilon_\theta = \frac{1}{r}\frac{\partial v}{\partial \theta} + \frac{u}{r}, \quad \gamma_{r\theta} = \frac{\partial v}{\partial r} + \frac{1}{r}\frac{\partial u}{\partial \theta} - \frac{v}{r} \tag{15.73}$$

where u and v are the displacement components in the r and θ directions, respectively.

Stress-Strain Relations

$$\begin{pmatrix} \varepsilon_r \\ \varepsilon_\theta \\ \gamma_{r\theta} \end{pmatrix} = \frac{1}{\tilde{E}} \begin{bmatrix} 1 & -\tilde{\mu} & 0 \\ -\tilde{\mu} & 1 & 0 \\ 0 & 0 & 2(1+\tilde{\mu}) \end{bmatrix} \begin{pmatrix} \sigma_r \\ \sigma_\theta \\ \tau_{r\theta} \end{pmatrix} \tag{15.74}$$

where $\tilde{E} = E, \tilde{\mu} = \mu$ for plane stress and $\tilde{E} = \frac{E}{1-\mu^2}, \tilde{\mu} = \frac{\mu}{1-\mu}$ for plane strain.

Condition of Compatibility

$$\frac{\partial^2 \varepsilon_\theta}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2 \varepsilon_r}{\partial \theta^2} + \frac{2}{r}\frac{\partial \varepsilon_\theta}{\partial r} - \frac{1}{r}\frac{\partial \varepsilon_r}{\partial r} = \frac{1}{r}\frac{\partial^2 \gamma_{r\theta}}{\partial r \partial \theta} + \frac{1}{r^2}\frac{\partial \gamma_{r\theta}}{\partial \theta} \tag{15.75}$$

Boundary Conditions Equations (15.61) in general can be applied here if the components in (x, y) are replaced by the components in (r, θ). However, in many cases, tractions and displacements are prescribed at $r =$ constant or $\theta =$ constant.

Stress Function For zero body forces, the equilibrium equations and the compatibility condition become

$$\frac{\partial \sigma_r}{\partial r} + \frac{1}{r}\frac{\partial \tau_{r\theta}}{\partial \theta} + \frac{\sigma_r - \sigma_\theta}{r} = 0, \quad \frac{1}{r}\frac{\partial \sigma_\theta}{\partial \theta} + \frac{\partial \tau_{r\theta}}{\partial r} + \frac{2\tau_{r\theta}}{r} = 0 \tag{15.76}$$

and

$$\nabla^2(\sigma_r + \sigma_\theta) = \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2}\right)(\sigma_r + \sigma_\theta) = 0 \tag{15.77}$$

Introduce a stress function Φ which is defined by

$$\sigma_r = \frac{1}{r}\frac{\partial \Phi}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \Phi}{\partial \theta^2}, \quad \sigma_\theta = \frac{\partial^2 \Phi}{\partial r^2}, \quad \tau_{r\theta} = -\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial \Phi}{\partial \theta}\right) \tag{15.78}$$

The stress components automatically satisfy the equilibrium equations (15.76). Substitute Eq. (15.78) into Eq. (15.77) to obtain the compatibility condition for the stress function

$$\nabla^4 \Phi = \left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2}\right)\left(\frac{\partial^2 \Phi}{\partial r^2} + \frac{1}{r}\frac{\partial \Phi}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \Phi}{\partial \theta^2}\right) = 0 \tag{15.79}$$

Thermal Stresses To determine the stresses caused by a temperature change $T(r, \theta)$, use the strain-stress relations

$$\begin{pmatrix} \varepsilon_r \\ \varepsilon_\theta \\ \gamma_{r\theta} \end{pmatrix} = \frac{1}{\tilde{E}} \begin{bmatrix} 1 & -\tilde{\mu} & 0 \\ -\tilde{\mu} & 1 & 0 \\ 0 & 0 & 2(1+\tilde{\mu}) \end{bmatrix} \begin{pmatrix} \sigma_r \\ \sigma_\theta \\ \tau_{r\theta} \end{pmatrix} + \alpha T \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \tag{15.80}$$

The compatibility condition becomes

$$\nabla^2(\sigma_r + \sigma_\theta + \alpha E T) = 0 \tag{15.81}$$

or

$$\nabla^2\left(\nabla^2 \Phi + \alpha E T\right) = 0 \tag{15.82}$$

where Φ is the stress function defined in Eq. (15.78).

Problems of Axial Symmetry If the geometric and material properties and loading of an elastic body are symmetric about the axis through the center of the polar coordinate system, the stress and displacement do not depend on θ and are functions of r only. The compatibility condition (15.79) in this case becomes

$$\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r}\right)\left(\frac{\partial^2 \Phi}{\partial r^2} + \frac{1}{r}\frac{\partial \Phi}{\partial r}\right) = 0 \tag{15.83}$$

and the stress function is

$$\Phi = c_1 r^2 + c_2 \log r \tag{15.84}$$

where constants c_1 and c_2 are determined by the boundary conditions. The stress, strain, and displacement components are given by

$$\sigma_r = \frac{1}{r}\frac{d\Phi}{dr} = 2c_1 + c_2\frac{1}{r^2}, \quad \sigma_\theta = \frac{\partial^2 \Phi}{\partial r^2} = 2c_1 - c_2\frac{1}{r^2}, \quad \tau_{r\theta} = 0 \tag{15.85}$$

$$\varepsilon_r = \frac{\partial u}{\partial r}, \quad \varepsilon_\theta = \frac{u}{r}, \quad \gamma_{r\theta} = \frac{\partial v}{\partial r} - \frac{v}{r} \tag{15.86}$$

$$u = \frac{1}{\tilde{E}}\left\{2c_1(1-\tilde{\mu})r - c_2(1+\tilde{\mu})\frac{1}{r}\right\}, \quad v = 0 \tag{15.87}$$

where $\tilde{E} = E, \tilde{\mu} = \mu$ for plane stress, and $\tilde{E} = \frac{E}{1-\mu^2}, \tilde{\mu} = \frac{\mu}{1-\mu}$ for plane strain. For plane strain, it has been assumed that the two ends of the body (along the z direction) are free to expand and that $\sigma_z = 0$.

EXAMPLE 3.3

The thick cylinder in Fig. 15.3.7 is subject to two sets of boundary conditions: (a) the cylinder is under uniform pressure on both the inner and outer surfaces; and (b) the cylinder is under uniform pressure on the inner surface, and is fixed on the outer surface. Due to the symmetry, the displacement component v in the θ direction is zero at every point.

Case (a): By the boundary conditions

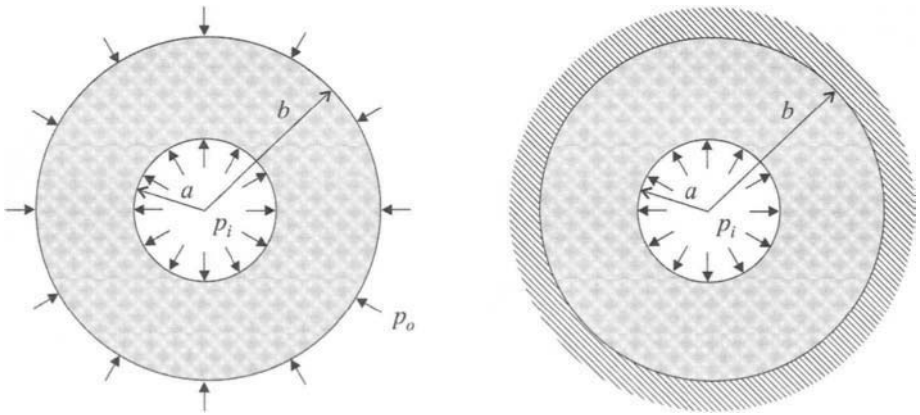
$$\sigma_r|_{r=a} = -p_i, \quad \sigma_r|_{r=b} = -p_o$$

the constants c_1 and c_2 in Eq. (15.85) are determined as

$$2c_1 = \frac{a^2p_i - b^2p_o}{b^2 - a^2}, \quad c_2 = \frac{a^2b^2(p_o - p_i)}{b^2 - a^2}.$$

Case (b): The boundary conditions

$$\sigma_r|_{r=a} = -p_i, \quad u|_{r=b} = 0$$



(a) (b)

FIGURE 15.3.7

leads to

$$2c_1 = -\frac{p_i(1+\tilde{\mu})a^2}{(1+\tilde{\mu})a^2+(1-\tilde{\mu})b^2}, \quad c_2 = -\frac{p_i(1-\tilde{\mu})a^2b^2}{(1+\tilde{\mu})a^2+(1-\tilde{\mu})b^2}$$

Substituting c_1 and c_2 into Eqs. (15.85) and (15.87) gives the stress and displacement solutions.

15.3.6 Stress Concentrations

Abrupt changes in geometry such as holes, notches, and cracks can cause large increases in stresses. In mechanics of materials, this effect of geometrical irregularities is known as *stress concentration*, and it is measured by the stress concentration factor

$$K_t = \frac{\sigma_{\max}}{\sigma_{\text{nom}}} \tag{15.88}$$

where $\sigma_{\max}$ is the maximum stress in the vicinity of the stress concentration and σ_{nom} is a nominal stress that would exist in the region of the question if the geometrical irregularity in consideration were absent.

EXAMPLE 3.4

One typical example of stress concentration is shown in Fig. 15.3.8, where a long plate with a small circular hole of radius a is under uniaxial tension σ_o. Assume that the width of the plate is much larger than the diameter of the hole, such that the edge effects are negligible. The stress function for the plate then is (Reference 4)

$$\Phi = -\frac{\sigma_o}{4}\left(r^2 + \frac{a^4}{r^2} - 2a^2\right)\cos 2\theta \quad \text{for } r \geq a \tag{15.89}$$

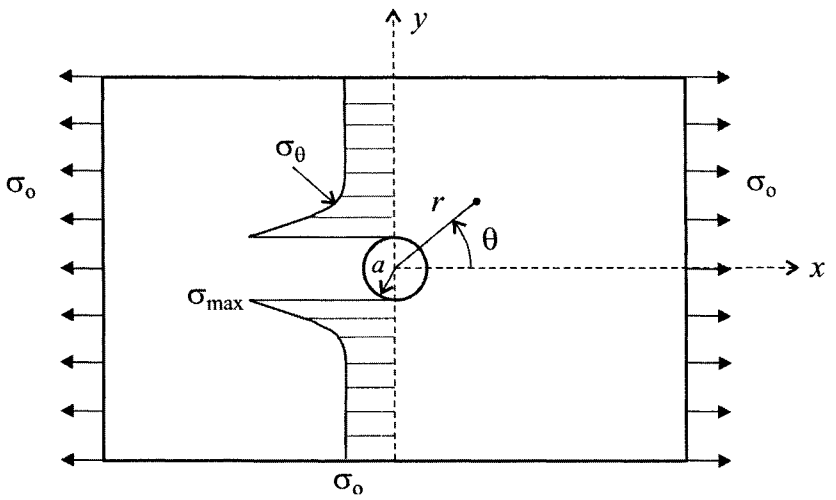


FIGURE 15.3.8

which, by Eq. (15.78), gives the stress components

$$\sigma_r = \frac{\sigma_o}{2}\left\{\left(1-\xi^2\right)+\left(1+3\xi^4-4\xi^2\right)\cos 2\theta\right\}$$
$$\sigma_\theta = \frac{\sigma_o}{2}\left\{\left(1+\xi^2\right)-\left(1+3\xi^4\right)\cos 2\theta\right\} \tag{15.90}$$
$$\tau_{r\theta} = -\frac{\sigma_o}{2}\left(1-3\xi^4+2\xi^2\right)\sin 2\theta$$

with $\xi = a/r$. The maximum stress occurs at the edge of the hole ($r = a$, $\theta = \pm\pi/2$), $\sigma_{\max} = \sigma_\theta = 3\sigma_o$. The nominal stress is $\sigma_{\text{nom}} = \sigma_o$. So, the stress concentration factor is

$$K_t = \frac{\sigma_{\max}}{\sigma_{\text{nom}}} = \frac{3\sigma_o}{\sigma_o} = 3.$$

15.4 Finite Element Method for 2-D Elasticity Problems

For most elasticity problems encountered in engineering practice, exact analytical solutions are difficult to obtain; numerical solutions are often sought instead. The finite element method (FEM) is the most powerful and versatile numerical method for various problems in solid mechanics and structural mechanics; see References 6 to 10 for instance.

In this section, a finite element formulation for two-dimensional (2-D) elasticity problems is presented, and a set of MATLAB functions for finite element analysis of elastic bodies is introduced. The reader who is familiar with the finite element method or simply wishes to use the MATLAB functions can skip Section 15.4.1 and directly go to Sections 15.4.2 and 15.4.3.

15.4.1 Finite Element Formulation

In the finite element method, an elastic body is divided into a number of subregions or finite elements, which are defined by nodes and boundaries connecting nodes. See Fig. 15.4.1, for instance, where a typical element (e) is defined by nodes i, j, and m, and straight-line boundaries. The body is assembled from those elements through interconnection at the nodes.

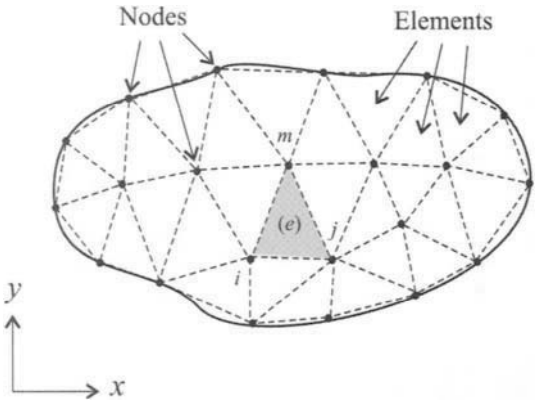


FIGURE 15.4.1 A body divided into a number of elements.

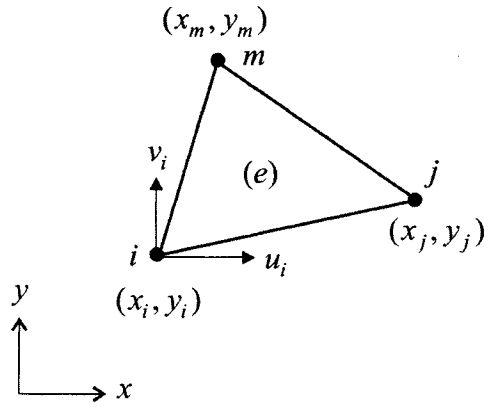


FIGURE 15.4.2 A triangular element for 2-D elasticity problems.

While finite elements can take different shapes, without loss of generality, only triangular elements are considered here.

A finite element analysis takes the following major steps.

Step 1. Displacement Interpolation A typical triangular element (e) for two-dimensional elasticity problems is shown in Fig. 15.4.2, where (x_i, y_i), etc. are the coordinates of the nodes i, j, and m, and $u_i = u(x_i, y_i), v_i = v(x_i, y_i)$, which are known as *nodal displacements*. Let the nodes be arranged in the counterclockwise direction. The displacements of the element are interpolated in the nodal displacements as follows

$$\{u_e\} = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix} = [N]\{q_e\} \tag{15.91}$$

where $\{q_e\}$ is a vector of nodal displacements

$$\{q_e\} = \begin{pmatrix} u_i & v_i & u_j & v_j & u_m & v_m \end{pmatrix}^T \tag{15.92}$$

and $[N]$ is a matrix of shape functions

$$[N] = \begin{bmatrix} N_i & 0 & N_j & 0 & N_m & 0 \\ 0 & N_i & 0 & N_j & 0 & N_m \end{bmatrix} \tag{15.93}$$

with

$$N_i = \frac{a_i + b_i x + c_i y}{2\Delta}, \quad N_j = \frac{a_j + b_j x + c_j y}{2\Delta}, \quad N_m = \frac{a_m + b_m x + c_m y}{2\Delta} \tag{15.94}$$

In Eq. (15.94),

$$\Delta = \frac{1}{2}\begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{vmatrix} \tag{15.95}$$

which equals the area of the triangle ijm; and

$$
\begin{aligned}
a_i &= \begin{vmatrix} x_j & y_j \\ x_m & y_m \end{vmatrix}, \quad b_i = y_j - y_m, \quad c_i = -x_j + x_m \\
a_j &= \begin{vmatrix} x_m & y_m \\ x_i & y_i \end{vmatrix}, \quad b_j = y_m - y_i, \quad c_j = -x_m + x_i \\
a_m &= \begin{vmatrix} x_i & y_i \\ x_j & y_j \end{vmatrix}, \quad b_m = y_i - y_j, \quad c_m = -x_i + x_j
\end{aligned} \tag{15.96}
$$

The shape function has the properties

$$
\begin{aligned}
N_i(x_i, y_i) &= 1, \quad N_i(x_j, y_j) = N_i(x_m, y_m) = 0 \\
N_j(x_j, y_j) &= 1, \quad N_j(x_m, y_m) = N_j(x_i, y_i) = 0 \\
N_m(x_m, y_m) &= 1, \quad N_m(x_i, y_i) = N_m(x_j, y_j) = 0
\end{aligned} \tag{15.97}
$$

and

$$
N_i + N_j + N_m = 1 \quad \text{at any point in element } (e) \tag{15.98}
$$

Figure 15.4.3 shows these properties. Note that N_i is zero at the boundary j-m and varies linearly along the other boundaries. This feature guarantees the continuity of the interpolated displacements on the boundary between any two adjacent elements.

Step 2. Formation of Element Stiffness Matrix and Nodal Force Vector The strain components of element (e), by Eqs. (15.56) and (15.91), are given by

$$
\{\varepsilon_e\} = \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix} = [B]\ \{q_e\} \tag{15.99}
$$

in which

$$
[B] = \frac{1}{2\Delta} \begin{bmatrix} b_i & 0 & b_j & 0 & b_m & 0 \\ 0 & c_i & 0 & c_j & 0 & c_m \\ c_i & b_i & c_j & b_j & c_m & b_m \end{bmatrix} \tag{15.100}
$$

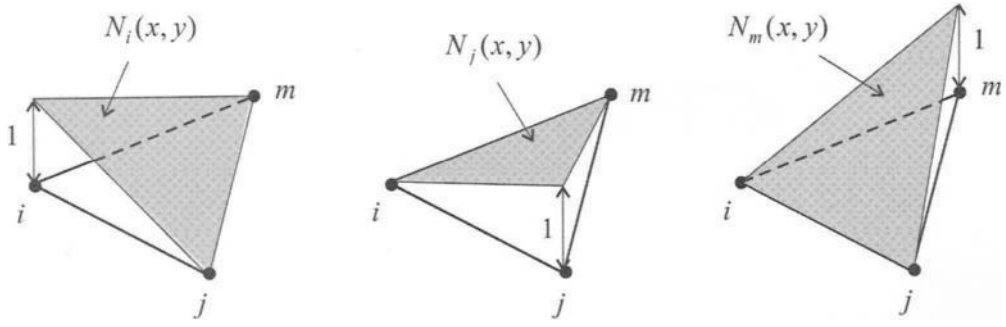


FIGURE 15.4.3 Shape functions.

Note that the strain of the element is constant because matrix $[B]$ is a constant matrix. The stress components of the element are of the form

$$\{\sigma_e\} = \begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} = [D]\{\varepsilon_e\} \tag{15.101}$$

where $[D]$ is an elasticity matrix of the form

$$[D] = \frac{\tilde{E}}{1-\tilde{\mu}^2}\begin{bmatrix} 1 & \tilde{\mu} & 0 \\ \tilde{\mu} & 1 & 0 \\ 0 & 0 & (1+\tilde{\mu})/2 \end{bmatrix} \tag{15.102}$$

with $\tilde{E} = E, \tilde{\mu} = \mu$ for plane stress, and $\tilde{E} = \frac{E}{1-\mu^2}, \tilde{\mu} = \frac{\mu}{1-\mu}$ for plane strain.

Let the region of element (e) be Ω_e, with boundary S_e. The potential energy of the element, according to Eqs. (15.43) and (15.99), is

$$\Pi_e = \frac{t}{2}\int_{\Omega_e} \{\varepsilon_e\}^T [D]\{\varepsilon_e\}\, dxdy - \int_{\Omega_e} \{u_e\}^T \begin{pmatrix} F_x \\ F_y \end{pmatrix} dxdy - \int_{S_e} \{u_e\}^T \begin{pmatrix} \bar{p}_x \\ \bar{p}_y \end{pmatrix} dS \tag{15.103}$$

where t is the thickness of the element, F_x and F_y are body forces (forces per unit area), and $\bar{p}_x$ and $\bar{p}_y$ are prescribed boundary tractions (forces per unit length). Substituting Eq. (15.91) into Eq. (15.103) gives

$$\Pi_e = \frac{1}{2}\{q_e\}^T [K_e]\{q_e\} - \{q_e\}^T \{f_e\} \tag{15.104}$$

where $[K_e]$, which is called the *stiffness matrix* of the element, is given by

$$[K_e] = t\Delta [B]^T [D][B] \tag{15.105}$$

and $\{f_e\}$, which is known as the *nodal force vector*, is given by

$$\{f_e\} = \int_{\Omega_e} [N]^T \begin{pmatrix} F_x \\ F_y \end{pmatrix} dxdy + \int_{S_e} [N]^T \begin{pmatrix} \bar{p}_x \\ \bar{p}_y \end{pmatrix} dS \tag{15.106}$$

The first term in Eq. (15.106) gives the nodal forces due to body forces, while the second term gives the nodal forces due to boundary tractions. For constant body forces, the integral

$$\int_{\Omega_e} [N]^T \begin{pmatrix} F_x \\ F_y \end{pmatrix} dxdy = \frac{\Delta}{3}\begin{pmatrix} F_x & F_y & F_x & F_y & F_x & F_y \end{pmatrix}^T \tag{15.107}$$

The second term in Eq. (15.106) involves the integration of the shape functions over the boundaries of the element. These functions vary linearly along a boundary of the element. For instance, on boundary ij (Fig. 15.4.4)

$$N_i = 1 - \frac{s}{h}, \quad N_j = \frac{s}{h}, \quad N_m = 0 \quad \text{for } 0 \le s \le h \tag{15.108}$$

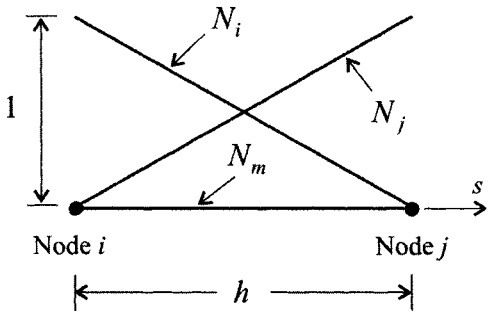


FIGURE 15.4.4 Shape functions on boundary ij of the element in Fig. 15.4.2.

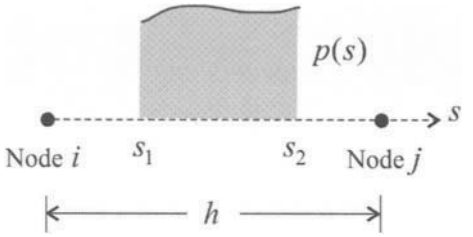


FIGURE 15.4.5 A traction distribution along boundary ij.

where s is a coordinate parameter along line ij and h is the distance between nodes i and j. So, for a traction distribution $p(s)$ defined between points s_1 and s_2 on boundary ij (Fig. 15.4.5)

$$\int_0^h N_i\, p(s)\, ds = R_0 - \frac{1}{h}R_1, \quad \int_0^h N_j\, p(s)\, ds = \frac{1}{h}R_0, \quad \int_0^h N_m\, p(s)\, ds = 0 \tag{15.109}$$

where

$$R_0 = \int_{s_1}^{s_2} p(s)\, ds, \quad R_1 = \int_{s_1}^{s_2} s\, p(s)\, ds \tag{15.110}$$

Based on Eqs. (15.106) and (15.109), the nodal forces due to $\bar{p}_x$ and $\bar{p}_y$ can be easily computed.

Step 3. Assembly of Entire Body Assume that the elastic body in consideration is divided into *Ne* finite elements. The body is assembled from these elements through reinforcement of force balance and displacement continuity at all the nodes. One convenient way to carry out this procedure is to make use of the Principle of Minimum Potential Energy (see Section 15.2.5). To this end, consider the potential energy of the entire body

$$\Pi = \sum_{e=1}^{Ne} \Pi_e = \sum_{e=1}^{Ne} \left(\frac{1}{2} \{q_e\}^T [K_e] \{q_e\} - \{q_e\}^T \{f_e\} \right) \tag{15.111}$$

where Eq. (15.104) has been used. Through introduction of boundary conditions, the above equation can be rewritten as

$$\Pi = \frac{1}{2}\{q\}^T [K_g]\{q\} - \{q\}^T \{f\} \tag{15.112}$$

where $[K_g]$, which is called the *global stiffness matrix*, is assembled from the element stiffness matrices $[K_e]$, vector $\{q\}$ contains all the independent nodal displacements, and $\{f\}$ contains the corresponding nodal forces. According to Eq. (15.47),

$$\delta\Pi = \delta\{q\}^T \left([K_g]\{q\} - \{f\}\right) = 0 \tag{15.113}$$

which, by the arbitrariness of $\delta\{q\}$, yields the global equilibrium equation of the body

$$[K_g]\{q\} = \{f\} \tag{15.114}$$

The nodal displacements satisfying the above equation will guarantee force balance and displacement continuity at the nodes.

Step 4. Finite Element Solution The finite element solution procedure for a 2-D elasticity problem is summarized as follows:

(a) Input the material and geometric data of the elastic body in consideration.
(b) Compute the element stiffness matrices $[K_e]$ according to Eq. (15.105).
(c) Compute the nodal force vectors $\{f_e\}$ by Eq. (15.106).
(d) Assemble the global stiffness matrix $[K_g]$ and nodal force vector $\{f\}$. (During this process, the boundary conditions and any constraints of the body are introduced.)
(e) Solve the equilibrium equation for the nodal displacements, namely,

$$\{q\} = [K_g]^{-1}\{f\} \tag{15.115}$$

(f) Substitute the nodal displacement solution into Eqs. (15.91), (15.99), and (15.101) to compute the displacement, strain, and stress at any point of the body.

The above solution procedure is illustrated in the following example.

EXAMPLE 4.1

In Fig. 15.4.6, a rectangular plate of thickness $t = 0.1$ m is subject to a pointwise load P. The plate is made of isotropic material with $E = 2{,}000$ Pa and $\mu = 0.3$. The plate is divided into two triangular elements. By Eq. (15.102), the elasticity matrix of the plate

$$[D] = \begin{bmatrix} 2197.8 & 659.3 & 0 \\ 659.3 & 2197.8 & 0 \\ 0 & 0 & 769.2 \end{bmatrix} \text{Pa.}$$

By Eq. (15.105), the stiffness matrices of the elements are obtained as follows:

for element (1)

$$[K_1] = \begin{bmatrix} 76.9231 & 0 & 0 & -38.4615 & -76.9231 & 38.4615 \\ 0 & 219.7802 & -32.9670 & 0 & 32.9670 & -219.7802 \\ 0 & -32.9670 & 54.9451 & 0 & -54.9451 & 32.9670 \\ -38.4615 & 0 & 0 & 19.2308 & 38.4615 & -19.2308 \\ -76.9231 & 32.9670 & -54.9451 & 38.4615 & 131.8681 & -71.4286 \\ 38.4615 & -219.7802 & 32.9670 & -19.2308 & -71.4286 & 239.0110 \end{bmatrix}$$

for element (2)

$$[K_2] = \begin{bmatrix} 54.9451 & 0 & -54.9451 & 32.9670 & 0 & -32.9670 \\ 0 & 19.2308 & 38.4615 & -19.2308 & -38.4615 & 0 \\ -54.9451 & 38.4615 & 131.8681 & -71.4286 & -76.9231 & 32.9670 \\ 32.9670 & -19.2308 & -71.4286 & 239.0110 & 38.4615 & -219.7802 \\ 0 & -38.4615 & -76.9231 & 38.4615 & 76.9231 & 0 \\ -32.9670 & 0 & 32.9670 & -219.7802 & 0 & 219.7802 \end{bmatrix}$$

The plate is fixed or clamped at nodes 1 and 2; namely $u_1 = v_1 = u_2 = v_2 = 0$. After introduction of these boundary conditions, the global stiffness matrix and nodal force vector in the equilibrium equation (15.114) are obtained as follows

$$[K_g] = \begin{bmatrix} 131.8681 & -71.4286 & -54.9451 & 38.4615 \\ -71.4286 & 239.0110 & 32.9670 & -19.2308 \\ -54.9451 & 32.9670 & 131.8681 & 0 \\ 38.4615 & -19.2308 & 0 & 239.0110 \end{bmatrix}, \quad \{f\} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -10 \end{pmatrix}$$

So, the displacements at nodes 3 and 4 are given by

$$\{q\} = \begin{pmatrix} u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix} = [K_g]^{-1}\{f\} = \begin{pmatrix} 0.015789 \\ 0.000251 \\ 0.006516 \\ -0.04436 \end{pmatrix} \text{m}.$$

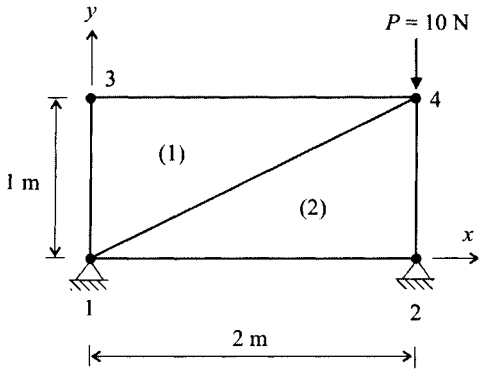


FIGURE 15.4.6

Substitute the computed nodal displacements into Eq. (15.101) to find the stress of each element as follows

$$\{\sigma_{(1)}\} = \begin{pmatrix} -10.0248 \\ -2.5062 \\ -5.0124 \end{pmatrix} \text{Pa}, \quad \{\sigma_{(2)}\} = \begin{pmatrix} -29.2481 \\ -97.4938 \\ 5.0124 \end{pmatrix} \text{Pa}.$$

15.4.2 MATLAB Solutions in Rectangular Regions

Consider a rectangular body of isotropic elastic material, with length a, height b, and thickness t. In a finite element analysis, the body is divided into one of the three M-by-N meshes of triangular elements, as shown in Fig. 15.4.7. In each mesh pattern, the grid widths along the x- and y-axes are

$$h_1 = \frac{a}{M}, \quad h_2 = \frac{b}{N} \tag{15.116}$$

and there are $2MN$ elements and $(M + 1) \times (N + 1)$ nodes.

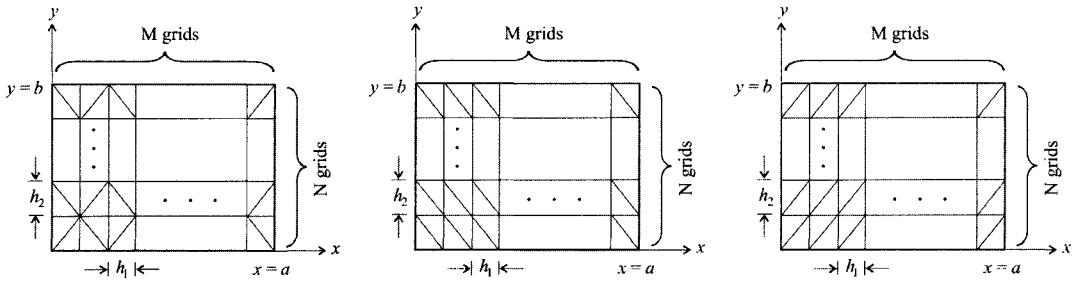


FIGURE 15.4.7 Three meshes of triangular elements for a rectangular body.

The Toolbox of this chapter provides several MATLAB functions for calculating finite element solutions of elasticity problems defined in rectangular regions. Solution by the Toolbox takes the following three steps:

Step 1. Set up a finite element mesh by function **setrectan**.
Step 2. Compute the static response of the elastic body by function **resprectan**.
Step 3. Display and plot the computed results by functions **eleminfo** and **plotmesh**.

These functions, along with others from the Toolbox, are presented in sequel.

Setup of Finite Element Mesh The finite element analysis by the Toolbox starts from setting up one of the meshes shown in Fig. 15.4.7. This is done by function `setrectan`; see Window 4.1.

The input arguments `BC_Spec` and `Constr_Spec` of `setrectan` are formed as follows:

(i) The boundary conditions of the body are assigned with a row vector

```
BC_Spec = [J1  J2  J3  J4]
```
(15.117)

Window 4.1. Function setrectan

MATLAB Function: **setrectan**

Purpose:
To set up parameters, boundary conditions, and a finite element mesh for a rectangular elastic body.

Synopsis:

```
setrectan(Properties, BC_Spec)
setrectan(Properties, BC_Spec, Constr_Spec, IS_Plane, Mesh_Indx)
```

Description:
`setrectan(Properties, BC)` specifies the material and geometric properties of the body in *plane stress* by a vector `Properties` and the boundary conditions by a vector `BC_Spec`. Here, `Properties = [E mu a b t]`, with `E` = Young's modulus, `mu` = Poisson's ratio, `a` = length, `b` = height, and `t` = thickness; and `BC_Spec` is formed according to Eq. (15.117) and Table 15.4.1. Also, `setrectan` establishes a 4-by-4 mesh of elements by default, and displays the information of the setup in the MATLAB command window.

`setrectan(Properties, BC_Spec, Constr_Spec, IS_Plane, Mesh_Indx)`, besides the previously mentioned jobs, also carries out the following tasks:

(i) It introduces constraints at selected nodes by a matrix `Constr_Spec`, whose formation is discussed after this window. If the body has no constraint, simply assign `Constr_Spec = 0`.

(ii) It clarifies the type of the elasticity problem by `IS_Plane` as below:

`IS_Plane = 1` for plane stress
`IS_Plane = 2` for plane strain

(iii) It specifies the finite element mesh by a vector

`Mesh_Indx = [M N Id_Pattern]`

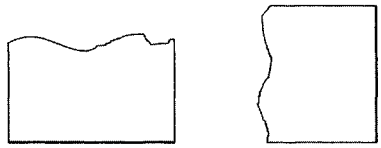
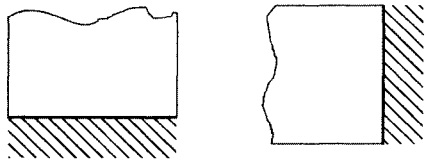
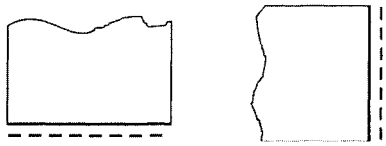
where `M` and `N` are the numbers of grids in the x- and y- directions, respectively, and `Id_Pattern` is an integer identifying the element mesh pattern by

`Id_Pattern = 1` for the mesh pattern in Fig. 15.4.7(a)
`Id_Pattern = 2` for the mesh pattern in Fig. 15.4.7(b)
`Id_Pattern = 3` for the mesh pattern in Fig. 15.4.7(c)

Also `Mesh_Indx = [M N]` has the same effect as `Mesh_Indx = [M N 1]`

By default, `BC_Spec = [0 0 0 1]`, `Constr_Spec = 0`, `IS_Plane = 1`, and `Mesh_Indx = [4 4]` for a 4-by-4 mesh of elements. The input arguments `BC_Spec`, `Constr_Spec`, `IS_Plane`, and `Mesh_Indx` can be replaced by `[ ]` as placeholder for their default values.

TABLE 15.4.1 Specification of Boundary Conditions

Jk(k = 1, 2, 3, 4)	*Examples of Boundaries*
Free edge Jk = 0	
Fixed edge ($u = v = 0$) Jk = 1	
Sliding edge ($u = 0$ or $v = 0$) Jk = 2	

where J1, J2, J3, and J4 are integers specifying the types of boundary conditions for the edges (B1), (B2), (B3), and (B4) of a body shown in Fig. 15.4.8, respectively. The boundaries of a body are labeled counterclockwise, starting from the bottom edge of the body; namely,

$$\begin{aligned} &\text{(B1)} \quad 0 \le x \le a, \quad y = 0 \\ &\text{(B2)} \quad x = a, \quad 0 \le y \le b \\ &\text{(B3)} \quad 0 \le x \le a, \quad y = b \\ &\text{(B4)} \quad x = 0, \quad 0 \le y \le b. \end{aligned} \tag{15.118}$$

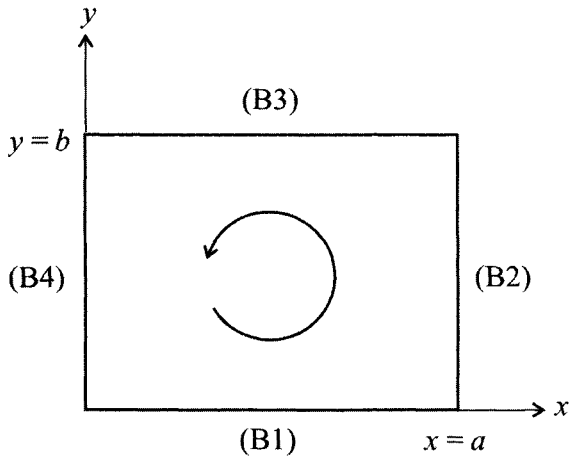


FIGURE 15.4.8 The labeled boundaries of a rectangular body.

For Jk = 0 or 1 or 2, a free or fixed or sliding edge is designated; see Table 15.4.1. For instance, BC_Spec = [1 2 1 0] specifies a body with two fixed edges (B1 and B3), one sliding edge (B2 with $u = 0$), and one free edge (B4).

(ii) Constraints can be imposed at specific nodes by a matrix

$$\texttt{Constr\_Spec} = \begin{bmatrix} \texttt{Constr\_1} \\ \texttt{Constr\_2} \\ \vdots \\ \texttt{Constr\_Nc} \end{bmatrix} \tag{15.119}$$

where Nc is the number of the constraints imposed. The jth row of the matrix has the form

$$\texttt{Constr\_j} = [\texttt{Node\_no} \quad \texttt{Type\_no}] \tag{15.120}$$

where Node_no is a node number and Type_no is a constraint type number with the following options:

Type_no = 1 for $u = 0$ at the specified node

Type_no = 2 for $v = 0$ at the specified node

Type_no = 3 for $u = 0$ and $v = 0$ at the specified node

Here u and v are nodal displacements. For example, Constr_Spec = [2 1; 14 3; 5 2] imposed three constraints as follows:

at node 2, $u = 0$;

at node 14, $u = v = 0$; and

at node 5, $v = 0$.

If an elastic body has no constraint at any node, simply assign Constr_Spec = 0. Also, pointwise constraints imposed by function setrectan can later on be changed or removed by calling function setconstr; see Window 4.2.

Window 4.2. Function setconstr

MATLAB Function: **setconstr**

Purpose:
To change or remove pointwise constraints on an elastic body.

Synopsis:

```
setconstr
setconstr(Constr_Spec)
```

Description:
setconstr removes all the pointwise constraints ever imposed on the elastic body.

Window 4.2. Function setconstr (*Continued*)

`setconstr(Constr_Spec)` imposes pointwise constrains by a matrix `Constr_Spec`, which is formed according to Eq. (15.119); and displays the constraints on an updated finite element mesh. The constraints introduced by `Constr_Spec` override any preexisting constraints.

EXAMPLE 4.2

The elastic body in Example 4.1 can be set up by

```
>> setrectan([2000 0.3 2 1 0.1], [0 0 0 0], [1 3; 2 3], 1, [1 1 3])
```

or

```
>> setrectan([2000 0.3 2 1 0.1], [1 0 0 0], 0, 1, [1 1 3])
```

The finite element mesh set up by functions `setrectan` and `setconstr` can be displayed at any time by the following command

```
>> dispFE
```

where `dispFE` is a MATLAB function from the Toolbox.

Computation of Static Response After a mesh of finite elements is established, the static response (displacements, strains, and stresses) of the body can be computed by function `resprectan`; see Window 4.3.

Window 4.3. Function resprectan

MATLAB Function: **resprectan**

Purpose:
To compute the static response of a rectangular body under external loads.

Synopsis:

```
resprectan(BT_Spec, PointF_Spec, BodyF_Spec)
[uu, ss] = resprectan(BD_Spec, PointF_Spec, BodyF_Spec)
```

Description:
`resprectan(BT_Spec, PointF_Spec, BodyF_Spec)` computes and displays the static response of the body under boundary tractions specified by a matrix `BT_Spec`, concentrated loads at nodes specified by a matrix `PointF_Spec`, and constant body forces specified by a vector `BodyF_Spec`. The formation of these input arguments is explained in what follows this window.

Window 4.3. Function resprectan (*Continued*)

If no boundary traction is applied, `BT_Spec = 0` or `[ ]`. Likewise `PointF_Spec = 0` or `[ ]` is for zero concentrated forces and `BodyF_Spec = 0` implies zero body forces. For example, `resprectan(BT_Spec)` computes the response of the body under boundary tractions only.

`[uu, ss] = resprectan(BD_Spec, PointF_Spec, BodyF_Spec)` returns the computed nodal displacements in a two-row matrix `uu`, and the strain and stress results in a ten-row matrix `ss`. The first row `uu(1,:)` of `uu` contains the nodal displacements u_k in the x-direction, while the second row `uu(2,:)` the nodal displacements v_k in the y-direction. The first three rows `ss(1:3, :)` of `ss` contain the strain components $\varepsilon_x, \varepsilon_y, \gamma_{xy}$ of the finite elements; the next three rows `ss(4:6, :)` the stress components $\sigma_x, \sigma_y, \tau_{xy}$; and the last four rows `ss(7:10, :)` the principal stresses and the their angles with respective the positive x-axis, $\sigma_1, \theta_{p1}, \sigma_2, \theta_{p2}$. The angles θ_{p1} and θ_{p2} are in degrees. For example, `uu(1,3)` gives the nodal displacement u_3; `ss(2,5)` the normal strain ε_y of the 5th element; `ss(6,10)` the shear stress τ_{xy} of the 10th element; and `ss(7, 8)` and `ss(8, 8)` the principal stress and associate angle σ_1, θ_{p1} of the 8th element.

Note:

Function `setrectan` must be called before `resprectan` can be used.

In use of `resprectan`, external forces are specified as follows.

(i) Boundary tractions are assigned by a matrix

$$\texttt{BT\_Spec} = \begin{bmatrix} \texttt{BT\_1} \\ \texttt{BT\_2} \\ \vdots \\ \texttt{BT\_Nt} \end{bmatrix} \tag{15.121}$$

where `Nt` is the number of the boundary tractions. The jth row of the matrix has the form

```
BT_j = [Bnd_ID  Force_Dir  Load_Type  z1  z2  par1  par2  par3]
```
(15.122)

where `Bnd_ID` is an integer identifying one of the four boundaries in Fig. 15.4.8 by

`Bnd_ID = k` for boundary (Bk), k = 1, 2, 3, 4

`Force_Dir` is an integer assigning the direction of the load by

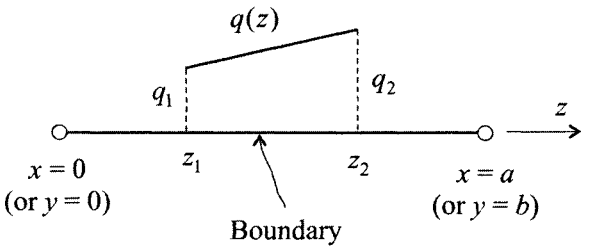
`Force_Dir = 1` for a load in the x-direction
`Force_Dir = 2` for a load in the y-direction

`Load_Type` is a load type number, `z1` and `z2` define the region of action of the load along the boundary, and `par1`, `par2`, and `par3` are parameters describing the distribution of the load by Table 15.4.2.

TABLE 15.4.2 Specification of Boundary Tractions

Traction	*Specification*

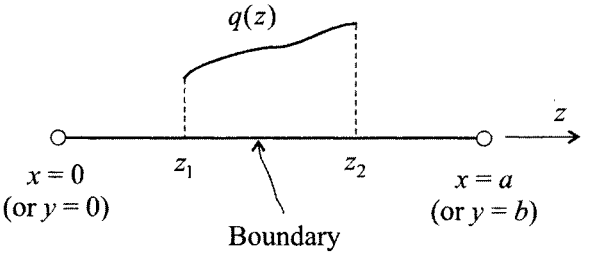
Load 1: Linear distribution



$$q(z) = q_1 \frac{z_2 - z}{z_2 - z_1} + q_2 \frac{z - z_1}{z_2 - z_1}, \quad z_1 \le z \le z_2$$

```
Load_Type = 1
par1 = q1
par2 = q2
par3 = 0
```

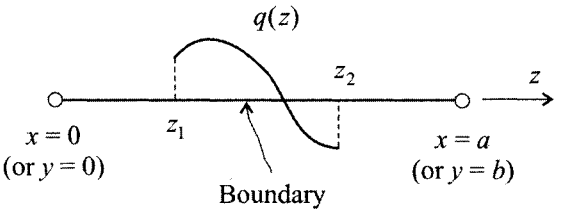
Load 2: Quadratic distribution



$$q(z) = a + bz + cz^2, \quad z_1 \le z \le z_2$$

```
Load_Type = 2
par1 = a
par2 = b
par3 = c
```

Load 3: Sinusoidal distribution



$q(z) = A \sin(\omega z + \theta_0), \quad z_1 \le z \le z_2$
(ω in rad per unit length, θ_0 in rad)

```
Load_Type = 3
par1 = A
par2 = ω
par3 = θ0
```

(ii) Concentrated loads at nodes are assigned by a matrix

$$\texttt{PointF\_Spec} = \begin{bmatrix} \texttt{PointF\_1} \\ \texttt{PointF\_2} \\ \vdots \\ \texttt{PointF\_Np} \end{bmatrix} \tag{15.123}$$

where `Np` is the number of nodes with concentrated loads, and the jth row of the matrix has the form

```
PointF_j = [node_no   px   py]
```  (15.124)

with `node_no` being a node number, and `px` and `py` being the x- and y-components of the load applied at the node.

(iii) A constant body force is assigned with a two-element vector

```
BodyF_Spec = [Fx  Fy]
```
(15.125)

where Fx and Fy are the x- and y-components of the body force.

EXAMPLE 4.3

In Fig. 15.4.9, a rectangular thin plate of thickness $t = 0.2$ m, which is fixed on its left edge and pinned at its lower-right corner ($v = 0$), is subject to two boundary tractions: a uniform load (q_0 = 7,000 N/m) on its top side, and a linearly distributed load (q_1 = 12,000 N/m) on its right side for 2m $\leq y \leq$ 4m. The elasticity constants of the body are $E = 20$ MPa and $\mu = 0.3$. Consider a 4-by-4 mesh (32 elements) for computing the finite element solution of the body. The command

```
>> setrectan([20e6 0.3 5 4 0.2], [0 0 0 1], [5 2])
```

specifies the material and geometric properties of the plate, assigns the boundary conditions and constraint, and sets up a finite element mesh (see Fig. 15.4.10). By

```
>> resprectan([2 1 1 2 4 0 -120000; 3 2 1 0 5 -70000 -70000])
```

the deformed configuration of the body is plotted in Fig. 15.4.11, and the nodal displacements are obtained as follows

```
Nodal Displacements:

   Node         u          v
   ----------------------------
   1.0000         0          0
   2.0000   -0.0189    -0.0404
   3.0000   -0.0172    -0.0681
```

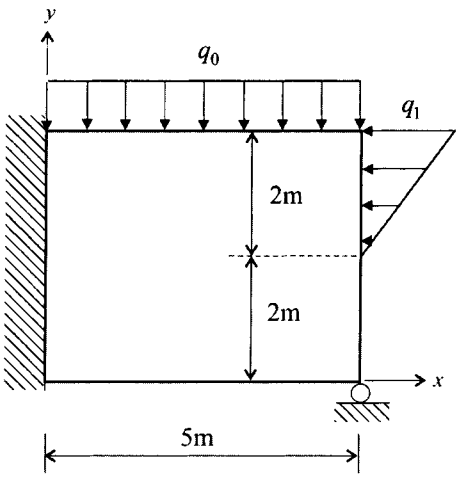


FIGURE 15.4.9

```
 4.0000   -0.0003   -0.0603
 5.0000    0.0086         0
 6.0000         0         0
 7.0000   -0.0105    0.0398
 8.0000   -0.0161   -0.0670
 9.0000   -0.0203   -0.0693
10.0000   -0.0090   -0.0345
11.0000         0         0
12.0000   -0.0087   -0.0419
13.0000   -0.0174   -0.0714
14.0000   -0.0235   -0.0768
15.0000   -0.0291   -0.0639
16.0000         0         0
17.0000   -0.0069   -0.0468
18.0000   -0.0145   -0.0783
19.0000   -0.0278   -0.0883
20.0000   -0.0369   -0.0848
21.0000         0         0
22.0000    0.0100   -0.0594
23.0000   -0.0009   -0.0901
24.0000   -0.0152   -0.1008
25.0000   -0.0209   -0.1019
```

which are in meters. The strain and stress components of each element can either be retrieved by

```
[uu, ss] = resprectan(BT_Spec)
```

as shown in Window 4.3, or be displayed by other functions to be introduced.

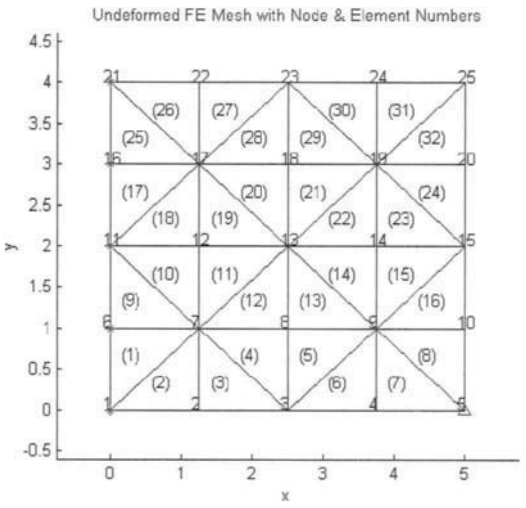


FIGURE 15.4.10 Undeformed finite element mesh.

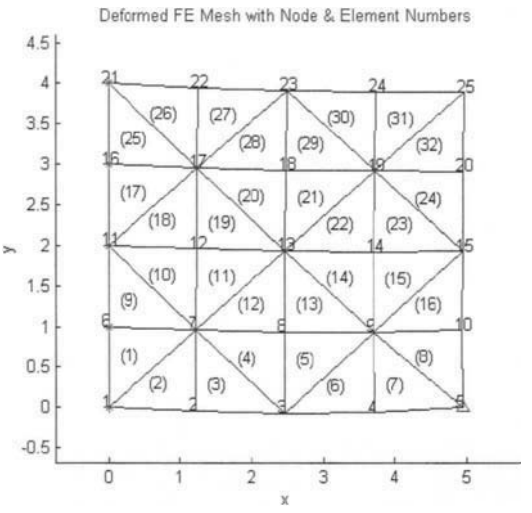


FIGURE 15.4.11 Deformed finite element mesh.

Presentation of Computed Results After the finite element solution of an elasticity problem is obtained by function `resprectan`, several MATLAB functions from the Toolbox can be used to display and retrieve the computed results. See Windows 4.4 to 4.8.

Displaying Response in Elements and at Nodes

Window 4.4. Function `eleminfo`

MATLAB Function: **`eleminfo`**

Purpose:
To display the information of a selected element of an elastic body.

Synopsis:

```
eleminfo(k)
Ke = eleminfo(k)
```

Description:
`eleminfo(k)` displays the nodes and stiffness matrix of the *k*th element. If the body is deformed, `eleminfo` also displays the nodal displacements, strain, and stress of the element.

`Ke = eleminfo(k)` returns the stiffness matrix of the *k*th element in a matrix `Ke`.

Window 4.5. Function nodeinfo

MATLAB Function: **nodeinfo**

Purpose:
To display the information of a selected node of an elastic body.

Synopsis:

```
nodeinfo(k)
[u, s, e] = nodeinfo(k)
```

Description:
`nodeinfo(k)` displays the coordinates of node `k`. If the body is deformed, `nodeinfo` also displays the displacements at the node, and the stress and strain that are averaged over all the elements interconnected at the node.

`[u, s, e] = nodeinfo(k)` returns the nodal displacements in a vector `u`, the averaged stress components in a vector `s`, and the averaged strain components in a vector `e`.

EXAMPLE 4.4

Consider Example 4.3. Assume that the static response has been computed by function `resprectan`.

```
>> eleminfo(20)
```

displays the static response about the 20th element as follows

```
Nodal displacements:
 at node 13: u = -0.035407, v = -0.064293
 at node 18: u = -0.026517, v = -0.071048
 at node 17: u = -0.013318, v = -0.040802

Strain components:
 normal strain Ex = -0.010559
 normal strain Ey = -0.0067548
 shear strain Lxy = -0.015307

Stress components:
 normal stress Sx = -276606.7537
 normal stress Sy = -218078.1143
 shear stress Txy = -117743.8264

>> nodeinfo(13)
```

gives the averaged strain and stress at node 13:

```
Averaged strain components:
 normal strain Ex = -0.013527
 normal strain Ey = -0.0044117
 shear strain Lxy = -0.0077112
Averaged stress components:
 normal stress Sx = -326380.9747
 normal stress Sy = -186147.9706
 shear stress Txy = -59317.1205
```

Plotting Finite Element Mesh

Window 4.6. Functions `plotmesh, plotmesh2`

MATLAB Functions: **`plotmesh, plotmesh2`**

Purpose:
To plot the finite element mesh of an elastic body in either undeformed or deformed configuration.

Synopsis:

```
plotmesh(opt_no)
plotmesh(opt_no, IS_Deform)
```

Description:
`plotmesh(opt_no)` plots a finite element mesh of an undeformed rectangular body that is set up by function `setrectan`, with the following options:

`opt_no = 0`	plot a mesh only
`opt_no = 1`	plot a mesh with node numbers
`opt_no = 2`	plot a mesh with element numbers
`opt_no = 3`	plot a mesh with both node and element numbers

Also, `plotmesh` has the same effect as `plotmesh(0)`.

`plotmesh(opt_no, IS_Deform)` plots a finite element mesh of the rectangular body in consideration, with `IS_Deform = 0` for an undeformed configuration and `IS_Deform = 1` for a deformed one.

`plotmesh2` has the same synopsis and functionality as `plotmesh`, except that it is for bodies of arbitrary shapes, as described in Section 15.4.3.

EXAMPLE 4.5

For the same elastic body in Example 4.3, use a 20-by-20 mesh (800 elements) for the finite element solution. The following commands

```
>> setrectan([20e6 0.3 5 4 0.2], [0 0 0 1], [21 2], 1, [20 20 2])
>> resprectan([2 1 1 2 4 0 -120000; 3 2 1 0 5 -70000 -70000])
>> plotmesh(0,1)
```

yield the deformed mesh of the body in Fig. 15.4.12. Here the mesh in Fig. 15.4.7(b) has been used.

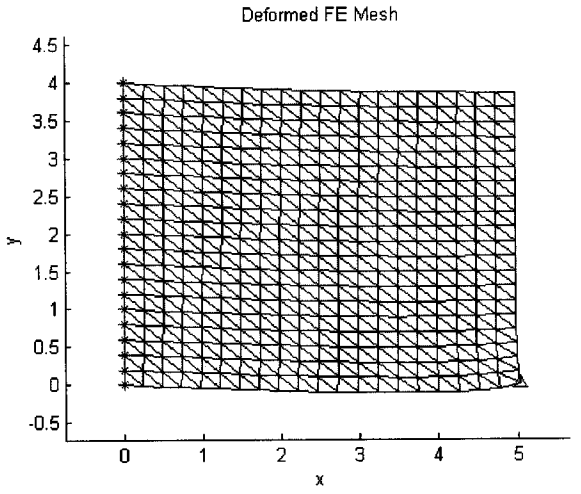


FIGURE 15.4.12 A 20-by-20 mesh of the deformed body.

Plotting Response Along Grid Lines An M-by-N mesh of a rectangular body consists of M+1 vertical grid lines (at $x = (j-1)\cdot h_1$, $j = 1, 2, \ldots,$ M+1) and N+1 horizontal grid lines (at $y = (k-1)\cdot h_2$, $k = 1, 2, \ldots,$ N+1), where h_1 and h_2 are the grid widths given in Eq. (15.116); see Fig. 15.4.13 for example. This Toolbox has a function `plotgln` for plotting the response of an elastic body along a grid line; see Window 4.7.

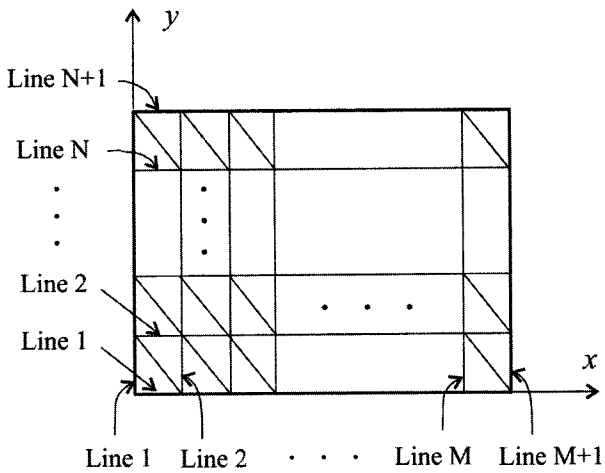


FIGURE 15.4.13 Grid lines of a finite element mesh.

Window 4.7. Function `plotgln`

MATLAB Function: **`plotgln`**

Purpose:
To plot the distributions of the response of a deformed elastic body along a grid line.

Synopsis:

```
plotgln(xy_dir, line_no)
[z,w,s] = plotgln(xy_dir, line_no)
```

Description:
`plotgln(xy_dir, line_no)` plot the spatial distributions of the displacement, strain, and stress of a deformed elastic body along a grid line of its finite element mesh. The input argument `xy_dir` is an integer identifying the direction of the response distribution as follows

`xy_dir = 1` response distributions in the x-direction

`xy_dir = 2` response distributions in the y-direction

and `line_no` is the grid line number as defined in Fig. 15.4.13.

`[z,w,s] = plotgln(xy_dir, line_no)` returns the x- or y-coordinates of those nodes on the selected grid line in a vector `z`, the displacements at the nodes in a two-row matrix `w`, and stress and strain components at the nodes in a six-row matrix `s`.

EXAMPLE 4.6

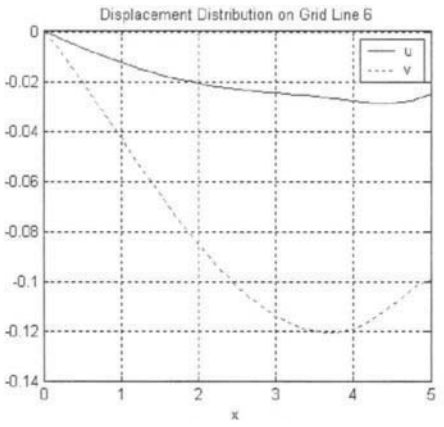
For the elastic body in Fig. 15.4.9, consider a 40-by-20 finite element mesh. The distributions of the displacement, strain, and stress of the body along the horizontal grid line at $y = 1$ m (in the x-direction) are plotted by

```
>> setrectan([20e6 0.3 5 4 0.2], [0 0 0 1], [41 2], 1, [40 20 2])
>> resprectan([2 1 1 2 4 0 -120000; 3 2 1 0 5 -70000 -70000])
>> plotgln(1,6)
```

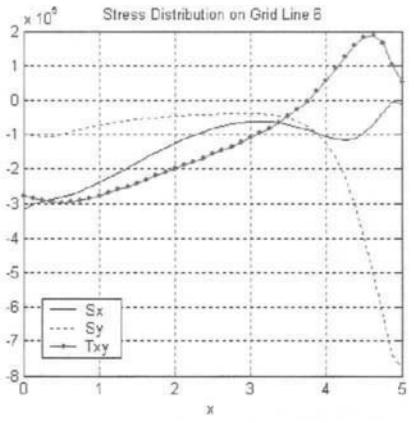
which yields Fig. 15.4.14. Also,

```
>> plotgln(2,21)
```

plots the displacement and stress distributions along the vertical grid line at $x =$ 2.5 m (in the y-direction) in Fig. 15.4.15.

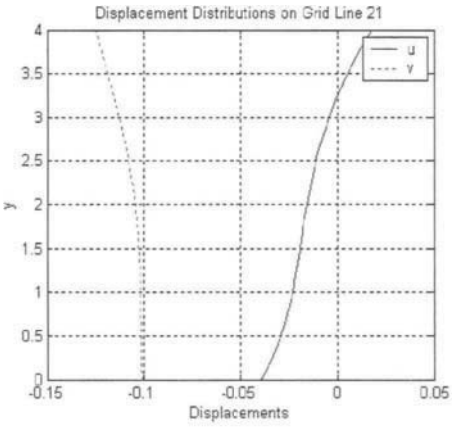


(a) Displacement distributions

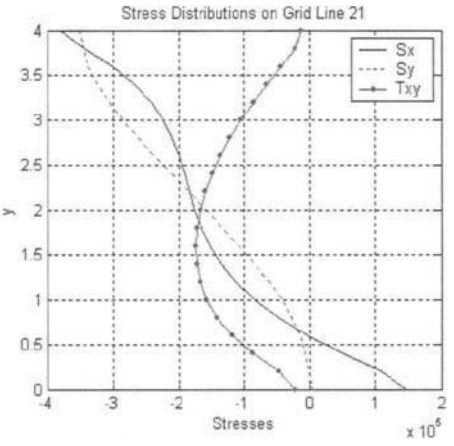


(b) Stress distributions

FIGURE 15.4.14 Displacement and stress distributions along the grid line at $y = 1$ m.



(a) Displacement distributions



(b) Stress distributions

FIGURE 15.4.15 Displacement and stress distributions along the grid line at $x = 2.5$ m.

Retrieving Global Stiffness Matrix

Window 4.8. Function `getKg`

MATLAB Function: **`getKg`**

Purpose:
To display and retrieve the global stiffness matrix of an elastic body.

Window 4.8. Function getKg (*Continued*)

Synopsis:

```
getKg(opt)
y = getKg(opt)
```

Description:

`getKg(opt)` displays the global stiffness matrix of an elastic body with the options:

`opt = 0` with no boundary conditions/constrains imposed
`opt = 1` with boundary conditions/constrains introduced

Also, `getKg` has the same effect as `getKg(0)`.

`y = getKg(opt)` returns the global stiffness matrix in a matrix *y*.

EXAMPLE 4.7

Consider the elastic body in Example 4.1.

```
>> setrectan([2000 0.3 2 1 0.1], [1 0 0 0], 0, 1, [1 1 3])
>> getKg
```

yields

```
Global Stiffness Matrix of the Elastic Body
-------------------------------------------
 No boundary conditions and constraints introduced
 [Kg] =
 131.8681         0  -54.9451   32.9670  -76.9231   38.4615         0  -71.4286
        0  239.0110   38.4615  -19.2308   32.9670 -219.7802  -71.4286         0
 -54.9451   38.4615  131.8681  -71.4286         0         0  -76.9231   32.9670
  32.9670  -19.2308  -71.4286  239.0110         0         0   38.4615 -219.7802
 -76.9231   32.9670         0         0  131.8681  -71.4286  -54.9451   38.4615
  38.4615 -219.7802         0         0  -71.4286  239.0110   32.9670  -19.2308
        0  -71.4286  -76.9231   38.4615  -54.9451   32.9670  131.8681         0
 -71.4286         0   32.9670 -219.7802   38.4615  -19.2308         0  239.0110
```

which is the global stiffness matrix before the boundary conditions are assigned. Also, `getKg(1)` retrieves a 4-by-4 stiffness matrix with the boundary conditions introduced, which is the same as shown in Example 4.1.

15.4.3 MATLAB Solutions in Arbitrary-Shaped Regions

The Toolbox of this chapter has MATLAB functions for finite element analysis of 2-D elastic bodies of arbitrary shapes. Solution by the Toolbox takes the following four steps:

Step 1. Establish a finite element mesh for the elastic body in consideration.
Step 2. Set up the elasticity problem by function **setabody**.

Step 3. Compute the static response of the elastic body by function **respabody**.
Step 4. Display and plot the computed results by functions **eleminfo** and **plotmesh2**.

These steps are explained in sequel.

Establishment of Finite Element Mesh A finite element mesh of an elastic body that has *Nn* nodes and *Ne* elements is described by the matrices Nodes_XY and Elem_Nodes:

$$\text{Nodes\_XY} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{Nn} & y_{Nn} \end{bmatrix}, \quad \text{Elem\_Nodes} = \begin{bmatrix} i_1 & j_1 & m_1 \\ i_2 & j_2 & m_2 \\ \vdots & \vdots & \vdots \\ i_{Ne} & j_{Ne} & m_{Ne} \end{bmatrix} \tag{15.126}$$

where x_j and y_j are the coordinates of the jth node; i_k, j_k, and m_k are the nodes of the kth element. For demonstrative purposes, consider the elastic body in Fig. 15.4.16(a). A mesh of triangular elements for the body is given in Fig. 15.4.16(b), where the numbers in parentheses are the element numbers while the numbers near nodes are node numbers. For this mesh

$$\text{Nodes\_XY} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 2 & 0 \\ 0 & 1 \\ 1 & 1 \\ 2 & 1 \\ 0 & 2 \\ 1 & 2 \\ 2 & 2 \\ 0 & 3 \\ 1 & 3 \end{bmatrix}, \quad \text{Elem\_Nodes} = \begin{bmatrix} 1 & 5 & 4 \\ 1 & 2 & 5 \\ 2 & 6 & 5 \\ 2 & 3 & 6 \\ 4 & 8 & 7 \\ 4 & 5 & 8 \\ 5 & 9 & 8 \\ 5 & 6 & 9 \\ 7 & 11 & 10 \\ 7 & 8 & 11 \end{bmatrix}$$

See Window 4.12 for automatic mesh generation for certain shapes.

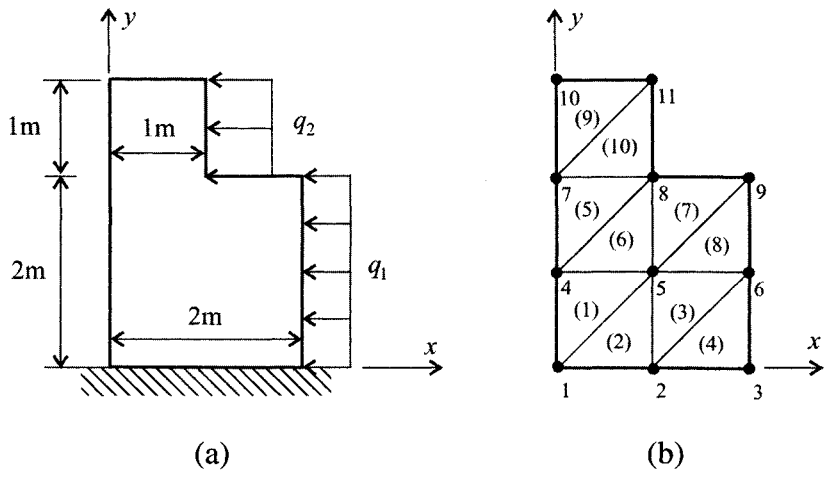


FIGURE 15.4.16 An elastic body and a mesh of elements for it.

Setup of Elasticity Problem In a finite element analysis by the Toolbox, the elastic body is set up by function `setabody`; see Window 4.9. Also, the boundary conditions of the body can be removed or changed by function `setbnd`; see Window 4.10.

Window 4.9. Function setabody

MATLAB Function: **setabody**

Purpose:
To assign parameters, boundary conditions, and a finite element mesh for an elastic body of arbitrary shape.

Synopsis:

```
setabody(Properties, Nodes_XY, Elem_Nodes)
setabody(Properties, Nodes_XY, Elem_Nodes, BC_Spec, IS_Plane)
```

Description:
`setabody(Properties, Nodes_XY, Elem_Nodes)` specifies the parameters of the body in *plane stress* by a vector `Properties = [E mu t]`, in which `E` = Young's modulus, `mu` = Poisson's ratio, and `t` = the thickness of the body, and forms a finite element mesh by matrices `Nodes_XY` and `Elem_Nodes`, which are assigned according to Eq. (15.126). In this setup, the boundary conditions of the body are not assigned. The boundary conditions can be specified by function `setbnd` (see Window 4.10).

`setabody(Properties, Nodes_XY, Elem_Nodes, BC_Spec, IS_Plane)`, besides specifying the parameters of the body and forming a finite element mesh, also carries out the following two tasks.

Task 1. It assigns the boundary conditions of the body as point constraints by a matrix

$$\texttt{BC\_Spec} = \begin{bmatrix} \texttt{BC\_1} \\ \texttt{BC\_1} \\ \vdots \\ \texttt{BC\_Nb} \end{bmatrix}$$

where `Nb` is the number of constraints, and the kth row of the matrix is of the form

```
BC_k = [Node_no  Type_no]
```

with `Node_no` being a node number, and `Type_no` a constraint type number with the options:

`Type_no = 1`	for $u = 0$ at the specified node
`Type_no = 2`	for $v = 0$ at the specified node
`Type_no = 3`	for $u = 0$ and $v = 0$ at the specified node

If no boundary conditions are to be assigned, set `BC_Spec = 0` or `[ ]`.

Task 2. It clarifies the type of the elasticity problem by `IS_Plane` as follows:

`IS_Plane = 1`	for plane stress
`IS_Plane = 2`	for plane strain

By default, `BC_Spec = 0` and `IS_Plane = 1`.

Window 4.10. Function setbnd

MATLAB Function: **setbnd**

Purpose:
To change or remove boundary conditions of an elastic body of arbitrary shape.

Synopsis:

```
setbnd
setbnd(BC_Spec)
```

Description:
`setbnd` removes all the boundary conditions ever assigned on the elastic body.

`setbnd(BC_Spec)` specifies boundary conditions as point constraints by a matrix `BC_Spec`, which is described in Window 4.9. The boundary conditions introduced by `BC_Spec` override any preexisting boundary conditions.

The finite element mesh set up by functions `setabody` and `setbnd` can be displayed at any time by the following command

```
>> dispFE2
```

where `dispFE2` is a MATLAB function from the Toolbox.

Computation of Static Response Once an elastic body has been set up by function `setabody`, the related elasticity problem can be solved by function `respabody`, as described in Window 4.11.

Window 4.11. Function respabody

MATLAB Function: **respabody**

Purpose:
To compute the static response of an arbitrary-shaped elastic body.

Synopsis:

```
respabody(PointF_Spec, BodyF_Spec)
[uu, ss] = respabody(PointF_Spec, BodyF_Spec)
```

Description:
`respabody(PointF_Spec, BodyF_Spec)` computes and displays the static response of the body under concentrated loads at nodes specified by a matrix `PointF_Spec`, and constant body forces specified by a vector `BodyF_Spec`. The formation of `PointF_Spec` and `BodyF_Spec` is the same as those for function `resprectan` (see Window 4.3). If no pointwise load is applied, `PointF_Spec = 0` or `[ ]`. On the other hand, for zero body forces, simply use `respabody(PointF_Spec)`.

Window 4.11. Function respabody (*Continued*)

`[uu, ss] = respabody(PointF_Spec, BodyF_Spec)` returns the computed nodal displacements in a two-row matrix `uu` and the strain and stress results in a ten-row matrix `ss`. The `uu` and `ss` are the same as those described in Window 4.3.

Note:

In use of `respabody`, boundary tractions are treated as pointwise loads according to Eqs. (15.106) and (15.109).

EXAMPLE 4.8

The elastic body in Fig. 15.4.16(a) is in plane stress, with the thickness $t = 0.15$ m, Young's modulus $E = 16$ MPa, and Poisson's ratio $\mu = 0.32$. Let the amplitudes of the boundary tractions shown in the figure be $q_1 = 6{,}000$ N/m and $q_2 = 9{,}000$ N/m. Consider the finite element mesh in Fig. 15.4.16(b). By Eqs. (15.106) and (15.109), the boundary tractions are treated as follows

at node 3, $p_x = -3{,}000$ N, $p_y = 0$
at node 6, $p_x = -6{,}000$ N, $p_y = 0$
at node 9, $p_x = -3{,}000$ N, $p_y = 0$
at node 8, $p_x = -4{,}500$ N, $p_y = 0$
at node 3, $p_x = -4{,}500$ N, $p_y = 0$.

The static response of the body is computed by

```
>> Nodes_XY = [0 0; 1 0; 2 0; 0 1; 1 1; 2 1; 0 2; 1 2; 2 2; 0 3; 1 3];
>> Elem_Nodes = [1 5 4; 1 2 5; 2 6 5; 2 3 6; 4 8 7; 4 5 8; 5 9 8;
                 5 6 9; 7 11 10; 7 8 11];
>> setabody([16e6 0.32 0.15], Nodes_XY, Elem_Nodes, [1 3; 2 3;
            3 3])
>> q1 = 6000; q2 = 9000;
>> Bnd_Tractions = [3 -q1/2 0; 6 -q1 0; 9 -q1/2 0; 8 -q2/2 0;
                    11 -q2/2 0];
>> respabody(Bnd_Tractions)
```

which yields the deformed mesh of the body in Fig. 15.4.17 and the nodal displacements as follows

```
Nodal Displacements:
  Node          u          v
 ----------------------------
   1.0000         0         0
   2.0000         0         0
   3.0000         0         0
   4.0000   -0.0143   -0.0088
   5.0000   -0.0133    0.0007
   6.0000   -0.0151    0.0076
```

```
 7.0000   -0.0314   -0.0133
 8.0000   -0.0307    0.0013
 9.0000   -0.0303    0.0108
10.0000   -0.0510   -0.0141
11.0000   -0.0518    0.0022
```

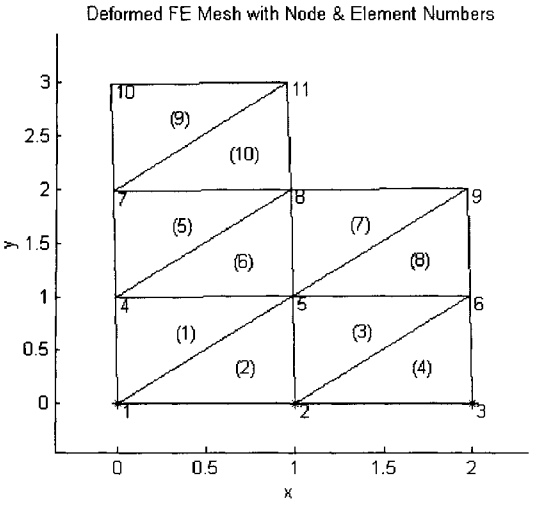


FIGURE 15.4.17

Presentation of Computed Results With the computed finite element results, functions `eleminfo` (Window 4.4) and `nodeinfo` (Window 4.5) can be used to show the static response in an element or at a node; function `plotmesh2` (Window 4.6) can be used to plot the finite element mesh in either undeformed or deformed configuration; and function `getKg` (Window 4.8) can be used to output the global stiffness matrix of the body. For instance, in Example 4.8

```
>> eleminfo(6)
```

retrieves the static response of the sixth element as follows

```
Nodal displacements:
 at node 4: u = -0.014284, v = -0.0088047
 at node 5: u = -0.013349, v =  0.00070685
 at node 8: u = -0.030667, v =  0.001288

Strain components:
 normal strain Ex =  0.00093562
 normal strain Ey =  0.00058114
 shear strain Lxy = -0.0078066

Stress components:
 normal stress Sx =  19992.553
 normal stress Sy =  15695.8491
 shear stress Txy = -47312.7953
```

Automatic Mesh Generation for Certain Shapes This Toolbox provides a function `getmesh` for automatic generation of finite element meshes for certain shapes; see Window 4.12.

Window 4.12. Function getmesh

MATLAB Function: **getmesh**

Purpose:
To generate a finite element mesh for one of the six shapes listed in Table 15.4.3.

Synopsis:

```
getmesh(Mesh_Spec, opt_no)
[Nodes_XY, Elem_Nodes] = getmesh(Mesh_Spec, opt_no)
```

Description:
`getmesh(Mesh_Spec, opt_no)` generates and displays a finite element mesh of a shape according to a row vector

```
Mesh_Spec = [Id_Pattern par1 par2 ...]
```

where `Id_Pattern` is an integer identifying the shape and mesh pattern and `par1` `par2` ... are the parameters describing the shape (see Table 15.4.3). Here `opt_no` is an integer of plotting options that is the same as for function `plotmesh2` (Window 4.6).

`[Nodes_XY, Elem_Nodes] = getmesh(Mesh_Spec, opt_no)` returns the coordinates of the nodes in a matrix `Nodes_XY` and the node numbers of the elements in a matrix `Elem_Nodes`. These matrices take the form given in Eq. (15.126), and can be used as the input arguments of function `setabody`.

Note 1. All the meshes listed in Table 15.4.3 are formed in the Cartesian coordinates (x, y). So, the nodal displacements and nodal forces used in computation must also be described in the Cartesian coordinates.

Note 2. For an annulus or annular sector listed in Table 15.4.3, a nonnegative parameter α is used to assign grids of varying width:

$$h_{k+1} = h_1(1 + k\alpha), \quad k = 1, 2, \ldots, M - 1$$

where h_k is the width of the kth grid or ring. Thus $\alpha = 0$ indicates grids of identical width. Also, when $\alpha = 0$,

$$\texttt{Mesh\_Spec} = [3 \quad R_{in} \quad R_{out} \quad M \quad N]$$

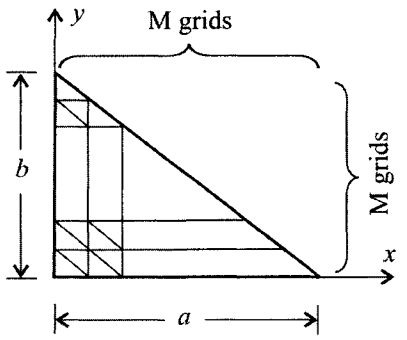
or

$$\texttt{Mesh\_Spec} = [4 \quad R_{in} \quad R_{out} \quad \theta \quad M \quad N]$$

can simply be used. Furthermore, the inner radius of an annulus or annular sector can be set to zero, i.e., $R_{in} = 0$. In that case, the annulus becomes a circle and the annular sector becomes a circular sector.

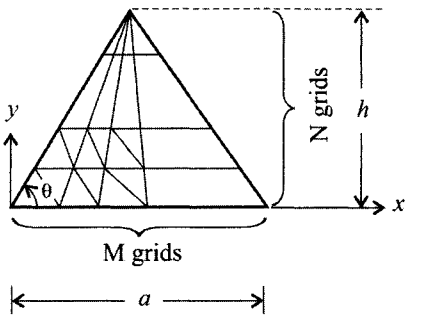
TABLE 15.4.3 Specification of Finite Element Meshes

Right triangle (Id_Pattern = 1)



Mesh_Spec = [1 a b M]

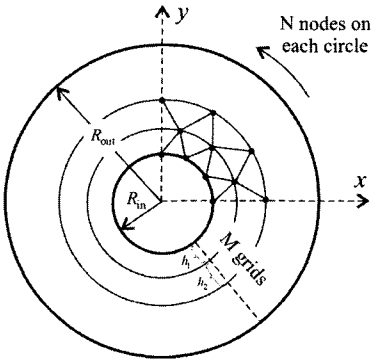
Oblique triangle (Id_Pattern = 2)



Mesh_Spec = [2 a h θ M N]

($0 < \theta < \pi/2$, θ in radians)

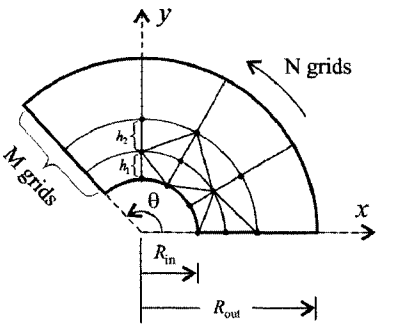
Annulus (Id_Pattern = 3)



Mesh_Spec = [3 R_{in} R_{out} M N α]

($\alpha \geq 0$)

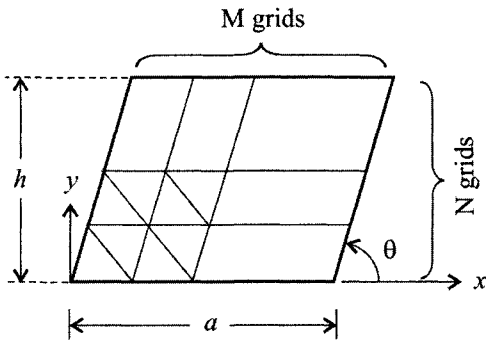
Annular sector (Id_Pattern = 4)



Mesh_Spec = [4 R_{in} R_{out} θ M N α]

($0 < \theta < 2\pi$, θ in radians; $\alpha \geq 0$)

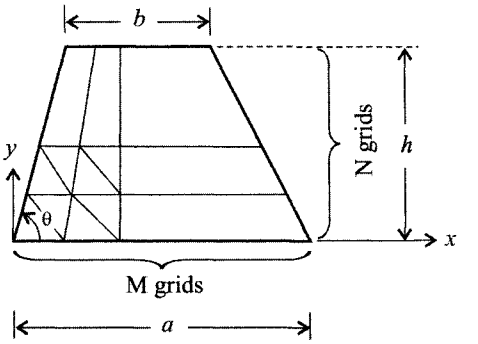
Parallelogram (Id_Pattern = 5)



Mesh_Spec = [5 a h θ M N]

($0 < \theta < \pi/2$, θ in radians)

Trapezoid (Id_Pattern = 6)



Mesh_Spec = [6 a b h θ M N]

($0 < \theta < \pi$, θ in radians)

EXAMPLE 4.9

In Fig. 15.4.18, a long cylinder is rigidly constrained on its lower-half outer surface, and is subject to a uniform load $q = 3 \times 10^6$ N/m along its longitudinal axis and through its diameter. The material and geometric parameters of the body are $E = 20$ MPa, $\mu = 0.28$, $a = 1.2$ m, and $b = 2.0$ m. To find the solution for this plane strain problem, take a segment of the cylinder of unit thickness ($t = 1$ m).

```
>> Mesh_Spec = [3 1.2 2 4 16];
>> [nodes, elements] = getmesh(Mesh_Spec,1);
```

sets up a 4-by-16 mesh (128 elements) with identical grid width; see Fig. 15.4.19. From the mesh, the fixed boundary conditions are specified at node 65 and nodes 73 to 80, and the location of the load q is identified as node 69; that is

```
>> BC_Spec = [[65 73:80]' 3*ones(9,1)];
>> PointF_Spec = [69 0 -3e6];
```

Thus,

```
>> E = 20e6; mu = 0.28; t = 1;
>> setabody([E mu t], nodes, elements, BC_Spec, 2)
>> respabody(PointF_Spec)
```

produces the deformed body plotted in Fig. 15.4.20. Also,

```
nodeinfo(69)
```

displays the displacements, averaged strain, and averaged stresses at node 69 (the location of the load) as follows.

```
Information about Node 69
-------------------------
 Coordinates: x = 0, y = 2

 Nodal displacements:
  u = 0
  v = -0.45201

 Averaged strain components:
  normal strain Ex = -0.069186
  normal strain Ey = -0.077815
  shear strain Lxy = 0

 Averaged stress components:
  normal stress Sx = -2542671.7547
  normal stress Sy = -2677500.5306
  shear stress Txy = 0
```

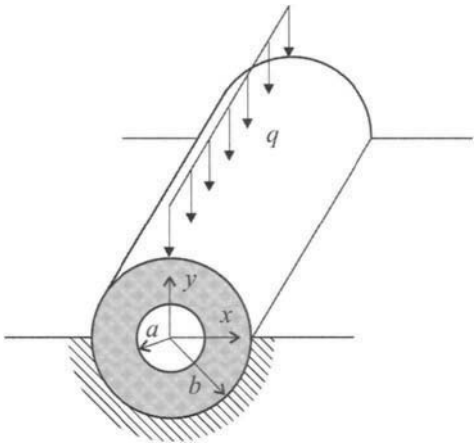


FIGURE 15.4.18 A long elastic cylinder.

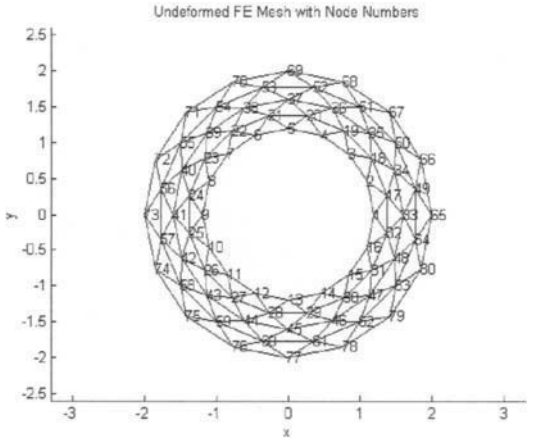


FIGURE 15.4.19 A finite element mesh for the cylinder.

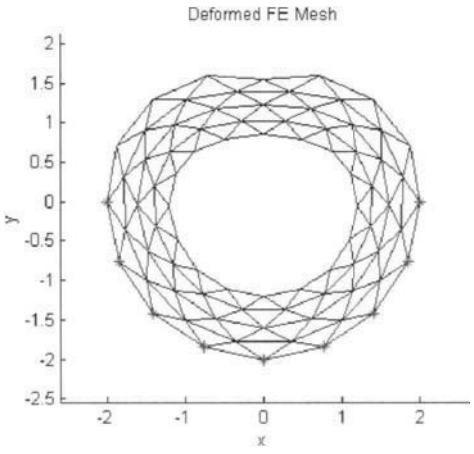


FIGURE 15.4.20 The deformed configuration of the body.

15.5 Quick Solution Guide

Governing Equations of an Elastic Body in Two Dimensions

Equilibrium Equations

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + F_x = 0$$
$$\frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + F_y = 0$$

Strain-Displacement Relations

$$\varepsilon_x = \frac{\partial u}{\partial x}, \quad \varepsilon_y = \frac{\partial v}{\partial y}, \quad \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$$

Stress-Strain Relations

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix} = \frac{\tilde{E}}{1-\tilde{\mu}^2} \begin{bmatrix} 1 & \tilde{\mu} & 0 \\ \tilde{\mu} & 1 & 0 \\ 0 & 0 & (1+\tilde{\mu})/2 \end{bmatrix} \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix}$$

where $\tilde{E} = E$, $\tilde{\mu} = \mu$ for plane stress, and $\tilde{E} = \frac{E}{1-\mu^2}$, $\tilde{\mu} = \frac{\mu}{1-\mu}$ for plane strain.

Condition of Compatibility

$$\frac{\partial^2 \varepsilon_x}{\partial y^2} + \frac{\partial^2 \varepsilon_y}{\partial x^2} = \frac{\partial^2 \gamma_{xy}}{\partial x \partial y}$$

Boundary Conditions

For prescribed tractions

$$\sigma_x l + \tau_{xy} m = \bar{p}_x$$
$$\tau_{xy} l + \sigma_y m = \bar{p}_y$$

For prescribed displacements

$$u = \bar{u}, \quad v = \bar{v}$$

MATLAB Functions

This chapter contains a set (toolbox) of MATLAB functions for static analysis of linearly elastic bodies, which are listed in the table below. Refer to the corresponding window or section for the utility of each function. The application of some functions is summarized in the following pages.

Stress and Strain		
`pristress3`	Compute principal stresses and principal axes in three dimensions	Window 2.1
`pristrain3`	Compute principal strains and principal axes in three dimensions	Window 2.2

Generalized Hooke's Law		
`hks2e`	Compute strain components based on Hooke's law	Window 2.3
`hke2s`	Compute stress components based on Hooke's law	Window 2.3
`elaconst`	Compute the values of elastic constants	Window 2.4
Finite Element Analysis of Two-Dimensional Bodies		
`Rectangular Regions`		
`setrectan`	Set up a finite element mesh for a rectangular elastic body	Window 4.1
`setconstr`	Change or remove pointwise constraints	Window 4.2
`resprectan`	Compute the static response of a rectangular elastic body	Window 4.3
`eleminfo`	Display the information of a selected element	Window 4.4
`nodeinfo`	Display the information at a selected node	Window 4.5
`plotmesh`	Plot the mesh of undeformed or deformed configuration	Window 4.6
`plotgln`	Plot static response of the body along a grid line	Window 4.7
`getKg`	Obtain global stiffness matrix	Window 4.8
`dispFE`	Display the information of a rectangular elastic body	Section 15.4
`Regions of Arbitrary Shapes`		
`setabody`	Set up a finite element mesh for an arbitrary-shaped body	Window 4.9
`setbnd`	Set up boundary conditions	Window 4.10
`respabody`	Compute the static response of a body	Window 4.11
`getmesh`	Obtain a finite element mesh for certain regions	Window 4.12
`eleminfo`	Display the information on a selected element	Window 4.4
`nodeinfo`	Display the information at a selected node	Window 4.5
`plotmesh2`	Plot the mesh for undeformed or deformed configuration	Window 4.6
`getKg`	Obtain global stiffness matrix	Window 4.8
`dispFE2`	Display the information of a body of arbitrary shape	Section 15.4
Utilities		
`TBdemo`	Show how the Toolbox works and what it can do	Section 15.1
`TBinfo`	Show the information of the Toolbox	Section 15.1
`RunEx`	Run all the numerical examples contained in this chapter	Section 15.1

Elasticity Solutions in Rectangular Regions Static analysis of a rectangular elastic body by the Toolbox takes the following three steps.

Step 1. Set up the rectangular body
To assign material and geometric parameters, boundary conditions, and finite element mesh for a rectangular elastic body, type

```
>> setrectan(Properties, BC_Spec, Constr_Spec, IS_Plane,
Mesh_Indx)
```

Step 2. Compute static response
To compute the displacements, strains, and stresses of the body subject to external loads, type

```
>> resprectan(BT_Spec, PointF_Spec, BodyF_Spec)
```

Step 3. Display results

To show the computed results about the jth element, type

```
>> eleminfo(j)
```

To show the computed results about the kth node, type

```
>> nodeinfo(k)
```

To plot the spatial distributions of the response of the body along a grid line, type

```
>> plotgln(xy_dir, line_no)
```

where `xy_dir = 1` for the x direction and `xy_dir = 2` for the y direction, and `line_no` is the grid line number.

Note. `setrectan`, `resprectan`, `eleminfo`, `nodeinfo`, and `plotgln` are functions from the Toolbox. For the assignment of the input arguments of function `setrectan`, see Window 4.1. For the assignment of the input arguments of function `resprectan`, see Window 4.2.

Elasticity Solutions in Regions of Arbitrary Shapes Static analysis of an elastic body of arbitrary shape takes the following four steps.

Step 1. Generate a finite element mesh

For one of the six types of regions listed in Table 15.4.3, type

```
>> [Nodes_XY, Elem_Nodes] = getmesh(Mesh_Spec);
```

Otherwise, generate the following matrices manually

$$\texttt{Nodes\_XY} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_{Nn} & y_{Nn} \end{bmatrix}, \quad \texttt{Elem\_Nodes} = \begin{bmatrix} i_1 & j_1 & m_1 \\ i_2 & j_2 & m_2 \\ \vdots & \vdots & \vdots \\ i_{Ne} & j_{Ne} & m_{Ne} \end{bmatrix}.$$

Step 2. Set up the elastic body

To assign material and geometric parameters, boundary conditions, and finite element mesh, type

```
>> setabody(Properties, Nodes_XY, Elem_Nodes, BC_Spec,
   IS_Plane)
```

where `Nodes_XY` and `Elem_Nodes` are obtained from Step 1.

Step 3. Compute static response

To compute the displacements, strains, and stresses of the body subject to external loads, type

```
>> respabody(PointF_Spec, BodyF_Spec)
```

Step 4. Display results

To show the computed results about the jth element, type

```
>> eleminfo(j)
```

To show the computed results about the kth node, type

```
>> nodeinfo(k)
```

Note. `getmesh`, `setabody`, and `respabody` are functions from the Toolbox. For the assignment of the input arguments of these functions, refer to Windows 4.9, 4.11, and 4.12.

15.6 References

References on Theory of Elasticity

1. Boresi A P, Chong K P 2000 *Elasticity in Engineering Mechanics*, 2nd edition, John Wiley & Sons, Inc.: New York.
2. Love A E 1994 *Treatise on the Mathematical Theory of Elasticity*, 4th edition, Dover Publications: New York.
3. Sokolnikoff I S 1956 *Mathematical Theory of Elasticity*, McGraw-Hill Book Company: New York.
4. Timoshenko S P, Goodier J N 1970 *Theory of Elasticity*, McGraw-Hill Book Company: New York.
5. Ugural A C, Fenster S K 2003 *Advanced Strength and Applied Elasticity*, 4th edition, Prentice-Hall, Inc.: Upper Saddle River, New Jersey.

References on Finite Element Methods

6. Cook R D, Malkus D S, Plesha M E, Witt R J 2001 *Concepts and Applications of Finite Element Analysis*, 4th edition, John Wiley & Sons, Inc.: New York.
7. Hughes T J R 2000 *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Publications: New York.
8. Kardestuncer H 1987 *Finite Element Handbook*, McGraw-Hill Book Company: New York.
9. Reddy J N 1993 *Introduction to the Finite Element Method*, 2nd edition, McGraw-Hill Book Company: New York.
10. Zienkiewicz O C, Taylor R L 2000 *The Finite Element Method*, 5th edition, Butterworth-Heinemann: Oxford.

This Page Intentionally Left Blank

16 Free Vibration of Membranes and Plates

Inside
- Getting Started
- Free Vibration of Membranes
- Free Vibration of Rectangular Plates
- Free Vibration of Circular Plates
- Quick Solution Guide
- References

16.1 Getting Started

What Is in This Chapter

This chapter is a package of subject review, fundamental theories, formulas, solution methods, and a set (toolbox) of MATLAB functions for free vibration analysis of tensioned membranes and thin plates, which are of rectangular and circular shapes.

System Requirements for the MATLAB Toolbox

- PC with Win 98/NT/2000 and XP, or Mac with OS 9.x and up
- The software MATLAB (version 5.x and up) installed

Software Installation and Test

(i) Drag the Toolbox folder from the CD onto a hard disk of your computer;
(ii) Launch MATLAB and set a path to the Toolbox folder on your hard disk;[1] and
(iii) Test the toolbox by typing **TBdemo** in the MATLAB command window, which will launch a demo program showing how the Toolbox works. The demo ends with a message: "The Toolbox works properly."

At this stage, the Toolbox has been properly installed, and it is ready for use.

[1] If the M-files of the Toolbox are put in the MATLAB work folder, there is no need to set a path.

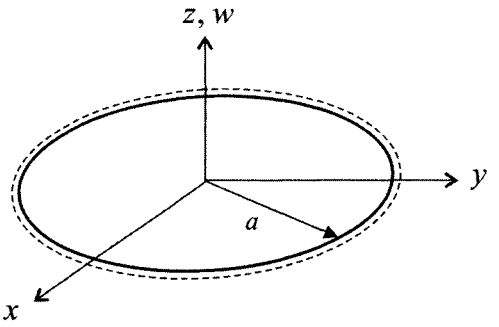


FIGURE 16.1.1 A simply-supported circular plate.

Quick Tutorial

To show how to use the MATLAB Toolbox, consider a simply-supported elastic circular plate in free vibration; see Fig. 16.1.1, where w is the transverse displacement of the plate. Let the plate parameters be: mass density $\rho = 2.6$ kg/m^2, radius $a = 0.45$ m, thickness $h = 0.01$ m, Young's modulus $E = 50,000$ Pa, Poisson's ratio $\mu = 0.3$. To compute the natural frequencies of the plate, type the following in the MATLAB command window:

```
>> rou = 2.6; E = 5e4; mu = 0.3;
>> a = 0.45; h = 0.01;
>> Plate_Pars = [rou E mu a h];
>> cirplate(Plate_Pars, 1)
```

where >> is the prompt in the MATLAB command window. This yields

```
Natural Frequencies:
--------------------
 w_mn       n = 0      1         2         3         4
-------------------------------------------------------------
m =
      0    1.0227    2.8802    5.3080    8.2805   11.7795
 1.0000    6.1590   10.0465   14.5307   19.5938   25.2209
 2.0000   15.3677   21.2982   27.8311   34.9553   42.6594
 3.0000   28.6643   36.6393   45.2191   54.3959   64.1613
 4.0000   46.0506   56.0706   66.6965   77.9227   89.7426

Here w_mn = natural frequency in rad/s
       m = number of nodal circles
       n = number of nodal diameters
```

The above natural frequencies are in rad/s. Furthermore,

```
>> plotmsh4(0, 1)
```

plots in Fig. 16.1.2 the spatial distribution of the plate displacement in mode (0,1) that has zero nodal circle and one nodal diameter, and displays the exact mathematical expression of the mode shape (eigenfunction) as follows

$$W_{01}(r,\theta) = [J_1(8.2845r) - 0.0055415I_1(8.2845r)]\cos\theta$$

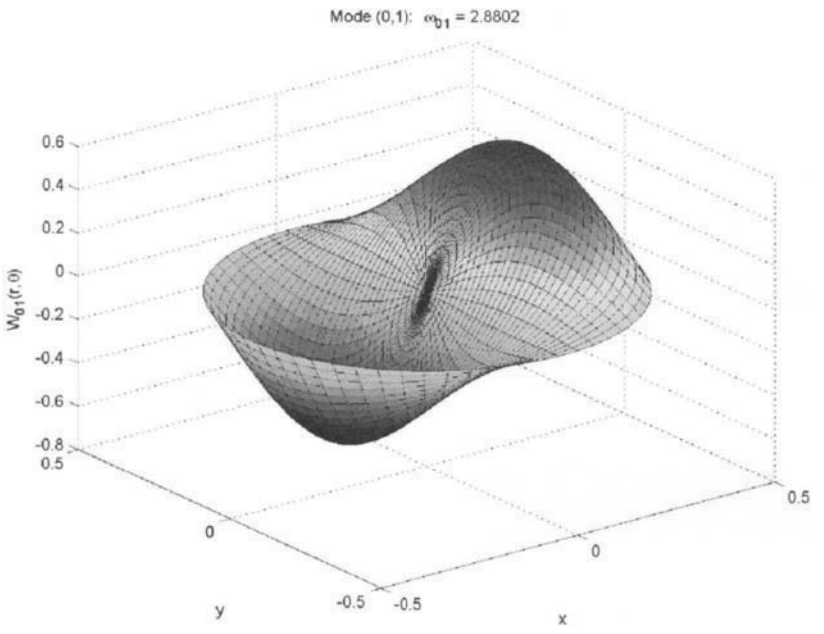


FIGURE 16.1.2 The spatial distribution of the plate displacement in mode (0,1).

where r and θ are the polar coordinates, and J_1 and I_1 are Bessel functions of the first and second kind, respectively.

In the above free vibration analysis of the plate, two MATLAB functions have been used:

(a) Function **`cirplate`** that computes the natural frequencies (eigenvalues) of the plate; and
(b) Function **`plotmsh4`** that displays a mode shape (eigenfunction) of the plate.

These functions and others from the Toolbox will be introduced in the subsequent sections.

For a quick solution, the user may refer directly to the Quick Solution Guide (Section 16.5), or get the information on the Toolbox by typing **`TBinfo`** in the MATLAB command window. To run the examples contained in this chapter, type **`RunEx`** in the MATLAB command window. To better understand how the Toolbox works, the user is encouraged to go through the entire chapter. For further reading on the subject, refer to Section 16.6.

16.2 Free Vibration of Membranes

The membrane in consideration is a thin unstretchable continuum that lies in a plane when in equilibrium, and is subject to uniform in-plane tension force along its boundary. The membrane has lateral or transverse vibration due to external and initial disturbances. Because the membrane has no resistance to bending, the elastic restoring force comes from the tension in the membrane, which is similar to that of a string in lateral vibration (see Chapter 13).

In what follows, free vibration of rectangular and circular membranes is presented, and MATLAB functions for solution of the related eigenvalue problems are presented.

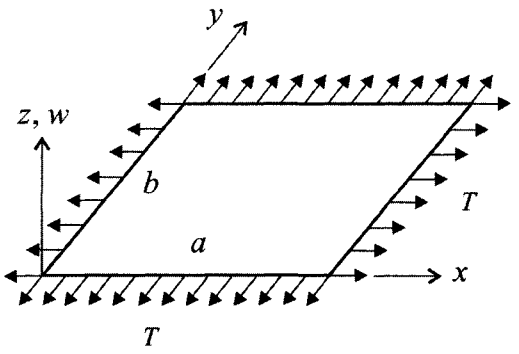


FIGURE 16.2.1 A rectangular membrane in transverse vibration.

16.2.1 Rectangular Membranes

Governing Equations In Fig. 16.2.1, a rectangular membrane of length a and width b is under a uniform tension T (force per unit length). The transverse displacement (lateral deflection) $w(x, y, t)$ of the membrane in free vibration is governed by the differential equation

$$T\nabla^2 w(x,y,t) = \rho \frac{\partial^2 w(x,y,t)}{\partial t^2}, \quad 0 \le x \le a, \quad 0 \le y \le b \tag{16.1}$$

where ρ is the mass density (mass per unit area), and the Laplacian operator

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{16.2}$$

Along the edges of the membrane, two types of boundary conditions are assigned:

Clamped Edges

$$\text{edges } x = 0,\ a \text{ or } y = 0, b \qquad w = 0 \tag{16.3}$$

Free Edges

$$\text{edge } x = 0, \text{ or } x = a \qquad \frac{\partial w}{\partial x} = 0 \tag{16.4a}$$

$$\text{edge } y = 0, \text{ or } y = b \qquad \frac{\partial w}{\partial y} = 0 \tag{16.4b}$$

Eigenvalue Problem A basic problem in free vibration of the membrane is to solve Eq. (16.1) subject to proper boundary conditions. To this end, express the membrane displacement as

$$w(x,y,t) = W(x,y)\cos\omega t \tag{16.5}$$

which, when substituted into Eq. (16.1), yields

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) W(x,y) + \frac{\rho\omega^2}{T} W(x,y) = 0 \tag{16.6}$$

Equation (16.6), along with the boundary conditions, defines an eigenvalue problem of the membrane, in which ω is an eigenvalue or natural frequency, and $W(x, y)$ is an eigenfunction or mode shape function.

The solution of Eq. (16.6), by separation of variables, is written as

$$W(x, y) = \Phi(x)\Psi(y) \tag{16.7}$$

Substitute Eq. (16.7) into Eq. (16.6) to obtain the differential equations

$$\frac{d^2\Phi}{dx^2} + \alpha^2\Phi = 0 \tag{16.8a}$$

and

$$\frac{d^2\Psi}{dy^2} + \beta^2\Psi = 0 \tag{16.8b}$$

where parameters α and β are related by

$$\alpha^2 + \beta^2 = \frac{\rho\omega^2}{T} \tag{16.9}$$

Equations (16.8) indicate that functions $\Phi(x)$ and $\Psi(y)$ are of sinusoidal form. It follows that the solution of Eq. (16.6) is

$$W(x, y) = A_1 \sin\alpha x \sin\beta y + A_2 \sin\alpha x \cos\beta y + A_3 \cos\alpha x \sin\beta y + A_4 \cos\alpha x \cos\beta y \tag{16.10}$$

where A_k are constants to be determined.

Equation (16.10), when used with the boundary conditions, leads to the eigensolutions (natural frequencies and mode shapes). As an example, consider a membrane that is clamped at edges $x = 0$, $x = a$, and $y = b$, and is free at edge $y = 0$. The boundary conditions of the membrane are

(i) At edge $x = 0$ $\quad W(0, y) = 0$

(ii) At edge $y = 0$ $\quad \left.\dfrac{\partial W(x, y)}{\partial y}\right|_{y=0} = 0$

(iii) At edge $x = a$ $\quad W(a, y) = 0$

(iv) At edge $y = b$ $\quad W(x, b) = 0$

Application of conditions (i) and (ii) to Eq. (16.10) gives $A_1 = A_3 = A_4 = 0$, and

$$W(x, y) = A_2 \sin\alpha x \cos\beta y$$

which, by conditions (iii) and (iv), leads to the characteristic equations

$$\sin\alpha a = 0, \quad \cos\beta b = 0.$$

The characteristic roots are

$$\alpha_m = \frac{m\pi}{a}, \qquad m = 1, 2, \ldots$$
$$\beta_n = \frac{(n - 1/2)\pi}{b}, \qquad n = 1, 2, \ldots$$

With the relation (16.9), the natural frequencies of the membrane are

$$\omega_{mn} = \pi \sqrt{\left(\frac{m}{a}\right)^2 + \left(\frac{n - 1/2}{b}\right)^2} \sqrt{\frac{T}{\rho}}, \quad m, n = 1, 2, \ldots$$

and the associate mode shapes are

$$W_{mn}(x, y) = \sin \alpha_m x \cos \beta_n y.$$

Solution by the Toolbox The Toolbox of this chapter provides a MATLAB function `recmemb` for computing eigensolutions of rectangular membranes; see Window 2.1.

Window 2.1. Function recmemb

MATLAB Function: **recmemb**

Purpose:
To compute the eigensolutions of a rectangular membrane.

Synopsis:

```
recmemb(Memb_Pars, BC_Spec)
recmemb(Memb_Pars, BC_Spec, mn_mode)
```

Description:
`recmemb(Memb_Pars, BC_Spec)` computes the natural frequencies and mode shapes of the membrane, where `Memb_Pars` is a vector of membrane parameters

```
Memb_Pars = [rou T a b]
```

with `rou` = mass density (mass per unit area), `T` = tension (force per unit length), `a` = length, and `b` = height; and `BC_Spec` is a vector of the form

```
BC_Spec = [J1 J2 J3 J4]
```

in which `Jk` is an integer specifying the boundary condition at the edge (`Bk`) of the membrane shown in Fig. 16.2.2, with the options

```
Jk = 0  for a free edge
Jk = 1  for a clamped edge
```

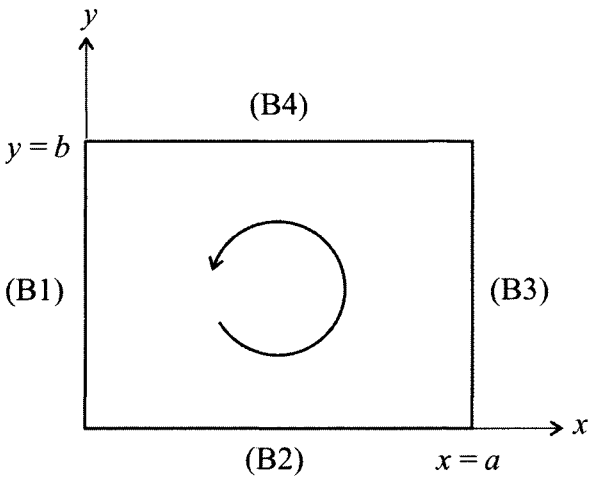


FIGURE 16.2.2 The labeled boundaries of a rectangular membrane.

Window 2.1. Function recmemb (*Continued*)

`recmemb(Memb_Pars, BC_Spec, mn_mode)` computes the eigensolutions of the membrane with `mn_mode = [m n]`, where `m` is the highest wave number of the mode shapes in the x direction and `n` is the highest wave number is the y direction. By default, `mn_mode = [4 4]`.

Note. After `recmemb` is executed, function **`systinfo`** can be called at any time, such as

```
>> systinfo
```

to get access to the information on the membrane and its eigensolutions.

EXAMPLE 2.1

Consider a rectangular membrane with clamped-free-clamped-free edges, and parameters

$$\rho = 0.04, \quad T = 30, \quad a = 2, \quad b = 1$$

The MATLAB commands

```
>> Memb_Pars = [0.02 30 2 1];
>> BC_Spec = [1 0 1 0];
>> recmemb(Memb_Pars, BC_Spec)
```

TABLE 16.2.1 Style Specification for Mode Shape Profile

IS_Color	*Style*	IS_Color	*Style*
0	Black mesh	[ ]	Same as IS_Color = 19
1	Red mesh	11	Red surface
2	Green mesh	12	Green surface
3	Blue mesh	13	Blue surface
4	Yellow mesh	14	Yellow surface
5	Magenta mesh	15	Magenta surface
6	Cyan mesh	16	Cyan surface
7	Copper mesh	17	Copper surface
8	Aquamarine mesh	18	Aquamarine surface
9	Mesh of color related to height	19	Surface of color related to height

give the natural frequencies of the membrane as follows

```
Natural Frequencies:
--------------------
 w_mn      n = 1       2        3        4
-----------------------------------------------
m =
 1.0000   60.8367  136.0350  250.8361  370.0551
 2.0000  121.6734  172.0721  272.0699  384.7649
 3.0000  182.5100  219.3498  304.1834  408.1049
 4.0000  243.3467  272.0699  344.1442  438.6995

Here  w_mn = natural frequency in rad/s
        m = wave number in x direction
        n = wave number in y direction
```

The Toolbox also has a function `plotmsh1` for plotting the spatial distribution of a particular mode shape, and a function `animodel` for animating the vibration of the membrane in a particular mode; see Windows 2.2 and 2.3.

Window 2.2. Functions `plotmsh1`, `plotmsh3`

MATLAB Functions: **`plotmsh1`, `plotmsh3`**

Purpose:
To plot a mode shape of a rectangular membrane or plate.

Synopsis:

```
plotmsh1(m, n)
plotmsh1(m, n, IS_Color, grid_pts)
[x, y, w] = plotmsh1(m, n, IS_Color, grid_pts)
```

Window 2.2. Functions plotmsh1, plotmsh3 (*Continued*)

Description:

`plotmsh1(m, n)` plots the spatial distribution of the displacement of a membrane in mode (m, n), shows the nodal lines of the mode on which $W_{mn} = 0$, and displays the analytical expression of the mode shape in the MATLAB command window. Here m and n are the wave numbers of the mode shape in the x and y directions, respectively.

`plotmsh1(m, n, IS_Color, grid_pts)` plots the mode shape with specified plotting control parameters, where `IS_Color` is an integer specifying the style of the mode shape (color, mesh, or surface) according to Table 16.2.1, and `grid_pts = [M N]` with `M, N` being integers specifying an `M`-by-`N` mesh of grid points in the membrane domain for the mode shape plot. By default, `IS_Color = 19`, and `M` and `N` are automatically selected according to the aspect (length-to-width) ratio of the membrane.

`[x, y, w] = plotmsh1(m, n, IS_Color, grid_pts)` returns matrices `x`, `y`, and `w` by which the three-dimensional profile of the mode shape can be generated by the MATLAB function `mesh(x, y, w)` or `surf(x, y, w)`. Here matrices `x` and `y` contain the coordinates of the grid points at which the modal displacement is computed, and matrix `w` contains the values of the modal displacement at the grid points.

The utility of function `plotmsh3` is the same as that of `plotmsh1`, except that `plotmsh3` is for a rectangular plate whose eigensolutions are obtained by Navier method or Lévy method (see Section 16.3.2).

Window 2.3. Functions animode1, animode3

MATLAB Functions: **animode1, animode3**

Purpose:

To animate a mode of vibration of a rectangular membrane or plate in its spatial domain.

Synopsis:

```
animode1(m, n)
animode1(m, n, IS_Color, grid_pts, tf, n_frames, IS_control)
F = animode1(m, n, IS_Color, grid_pts, tf, n_frames, IS_control)
```

Description:

`animode1(m, n)` animates the vibration of a rectangular membrane of mode `(m, n)`.

`animode1(m, n, IS_Color, grid_pts, tf, n_frames, IS_control)` animates the mode `(m, n)` of vibration with specified control parameters for animation, where `IS_Color` is an integer specifying the style of the modal displacement according to Table 16.2.1, `grid_pts = [M N]` specifying an `M`-by-`N` mesh of grid points for animation, `tf` is total animation time, `n_frames` is the number of frames in the animation, and `IS_control` is an animation control parameter with the following options:

`IS_control = 0`	play frames continuously
`IS_control = 1`	play frames one by one

Window 2.3. Functions `animode1`, `animode3` (*Continued*)

By default, `IS_Color = 19`, `M` and `N` are automatically selected, `tf = 8*T`, with T being the period of the mode, `n_frames = 97`, and `IS_control = 0`. Each of the above control parameters can be replaced by `[ ]` as a placeholder for its default value.

`F = animode1(m, n, IS_Color, grid_pts, tf, n_frames, IS_control)` returns a set of frames of membrane vibration in a matrix `F`, which can be played by the MATLAB function `movie(F)`.

The utility of function `animode3` is the same as that of `animode1`, except that `animode3` is for a rectangular plate whose eigensolutions are obtained by Navier method or Lévy method (see Section 16.3.2).

EXAMPLE 2.2

For the same membrane as in Example 2.1,

```
>> recmemb([0.02 30 2 1], [1 0 1 0])
>> animode1(2, 1, 14, [ ], 2*pi/121.6734, 9, 1)
```

animates the spatial profile of mode (2,1) with 9 frames in a total time $\texttt{tf} = 2\pi/\omega_{21} = 2\pi/121.6734$. The first six frames of the animation are shown in Fig. 16.2.3.

16.2.2 Circular Membranes

Governing Equations For a circular membrane of radius a, tension T, and density ρ (Fig. 16.2.4), its transverse displacement $w(r,\theta,t)$ in free vibration is governed by the differential equation

$$T\nabla^2 w(r,\theta,t) = \rho\frac{\partial^2 w(r,\theta,t)}{\partial t^2}, \quad 0 \le r \le a, \quad 0 \le \theta \le 2\pi \tag{16.11}$$

where r and θ are the polar coordinates, and the Laplacian operator

$$\nabla^2 = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} \tag{16.12}$$

Two typical types of boundary conditions at edge $r = a$ are as follows.

Clamped Edge

$$(w)_{r=a} = 0 \tag{16.13a}$$

Free Edge

$$\left(\frac{\partial w}{\partial r}\right)_{r=a} = 0 \tag{16.13b}$$

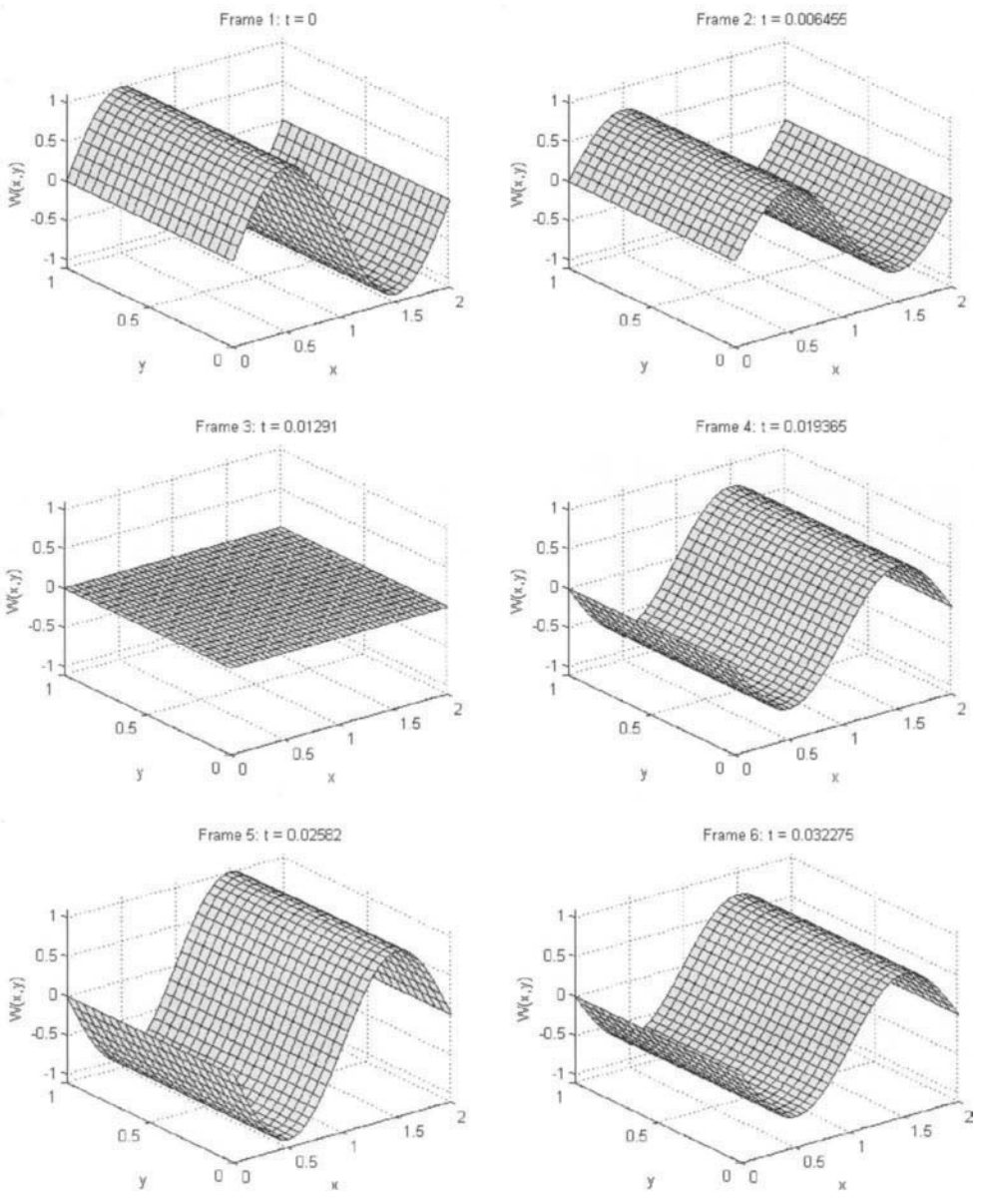


FIGURE 16.2.3 Frames of the animated membrane vibration in mode (2,1).

Eigensolutions One task in free vibration analysis of circular membranes is to solve Eq. (16.11) with one of the boundary conditions (16.13). Assume

$$w(r,\theta,t) = W(r,\theta)\cos\omega t \tag{16.14}$$

which, when substituted into Eq. (16.11), leads to

$$\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial\theta^2}\right) W(r,\theta) + \frac{\rho\omega^2}{T} W(r,\theta) = 0 \tag{16.15}$$

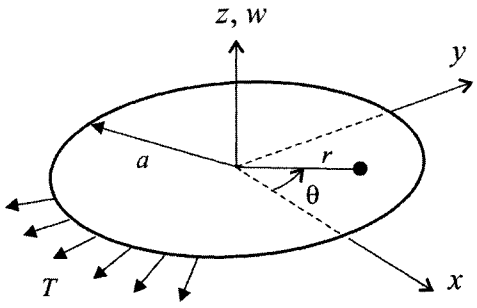


FIGURE 16.2.4 A circular membrane in transverse vibration.

where ω is an eigenvalue and $W(r,\theta)$ is the eigenfunction associated with ω. By separation of variables, write

$$W(r,\theta) = F(r)\cos n\theta \tag{16.16}$$

where n is an integer. This reduces Eq. (16.15) to

$$\frac{d^2F}{dr^2} + \frac{1}{r}\frac{dF}{dr} + \left(\gamma^2 - \frac{n^2}{r^2}\right)F = 0 \tag{16.17}$$

with

$$\gamma = \omega\sqrt{\rho/T} \tag{16.18}$$

Introduce the nondimensional variable

$$\xi = \gamma r \tag{16.19}$$

in Eq. (16.17) to obtain Bessel's equation

$$\frac{d^2\hat{F}}{d\xi^2} + \frac{1}{\xi}\frac{d\hat{F}}{d\xi} + \left(1 - \frac{n^2}{\xi^2}\right)\hat{F} = 0 \tag{16.20}$$

where $\hat{F}(\xi) = F(\xi/\gamma)$. A general solution of Eq. (16.20) is

$$\hat{F}(\xi) = AJ_n(\xi) + BY_n(\xi) = AJ_n(\gamma r) + BY_n(\gamma r) \tag{16.21}$$

where J_n and Y_n are Bessel functions of the first and second kind, and coefficients A, B are determined by boundary conditions. Because Y_n is infinite at $\xi = 0$, for a circular membrane without a central hole, B must be zero to avoid a singular deflection. Thus, the eigenfunction of the circular membrane is given by

$$W(r,\theta) = J_n(\gamma r)\cos n\theta. \tag{16.22}$$

TABLE 16.2.2 Roots $\gamma_{mn}a$ of Characteristic Equation (16.23)

	$n = 0$	1	2	3	4
$m = 0$	2.4048	3.8317	5.1356	6.3802	7.5883
1	5.5201	7.0156	8.4172	9.7610	11.0647
2	8.6537	10.1735	11.6198	13.0152	14.3725
3	11.7915	13.3237	14.7960	16.2235	17.6160
4	14.9309	16.4706	17.9598	19.4094	20.8269

Membrane with Clamped Boundary Substituting the expression (16.22) into the boundary condition (16.13a) gives the characteristic equation

$$J_n(\gamma a) = 0 \tag{16.23}$$

which has an infinite number of roots γ_{mn}, $m, n = 0, 1, 2, \ldots$ According to Eq. (16.18), these roots are related to the natural frequencies by

$$\omega_{mn} = \gamma_{mn}\sqrt{T/\rho} \tag{16.24}$$

The nondimensional roots $\gamma_{mn}a$ of Eq. (16.23) are given in Table 16.2.2. The mode shapes corresponding to γ_{mn} are given by

$$W_{mn}(r, \theta) = J_n(\gamma_{mn}r)\cos n\theta, \quad m, n = 0, 1, 2, \ldots \tag{16.25}$$

Setting $W_{mn}(r, \theta)$ to zero defines two types of nodal lines: nodal circles that are determined by $J_n(\gamma_{mn}r) = 0$, and nodal diameters that are determined by $\cos n\theta = 0$. The number of nodal circles excluding the membrane boundary is m; the number of nodal diameters is n.

Membrane with Free Boundary With Eq. (16.22), the boundary condition (16.13b) becomes

$$\left.\frac{dJ_n}{d\xi}\right|_{\xi=\gamma a} = 0 \tag{16.26}$$

which, by the identity

$$\xi\frac{dJ_n(\xi)}{d\xi} = nJ_n(\xi) - \xi J_{n+1}(\xi) \tag{16.27}$$

gives the characteristic equation

$$\frac{n}{\gamma a}J_n(\gamma a) - J_{n+1}(\gamma a) = 0 \tag{16.28}$$

The natural frequencies and mode shapes of the membrane thus can be determined from Eqs. (16.28) and (16.22). The roots of Eq. (16.28) are collected in Table 16.2.3, where the zero $\gamma_{00}a$ represents a rigid-body mode of the membrane.

TABLE 16.2.3 Roots $\gamma_{mn}a$ of Characteristic Equation (16.28)

	$n = 0$	1	2	3	4
$m = 0$	0	1.8412	3.0542	4.2012	5.3176
1	3.8317	5.3314	6.7061	8.0152	9.2824
2	7.0156	8.5363	9.9695	11.3459	12.6819
3	10.1735	11.7060	13.1704	14.5858	15.9641
4	13.3237	14.8636	16.3475	17.7887	19.1960

Solution by the Toolbox The Toolbox of this chapter provides a MATLAB function `cirmemb` for computing the eigensolutions of a circular membrane; see Window 2.4.

Window 2.4. Function `cirmemb`

MATLAB Function: **`cirmemb`**

Purpose:
To compute the natural frequencies and mode shapes of a circular membrane.

Synopsis:

```
cirmemb(Memb_Pars, BC_Spec)
cirmemb(Memb _Pars, BC_Spec, mn_mode)
```

Description:
`cirmemb(Memb_Pars, BC_Spec)` computes the natural frequencies and mode shapes of the membrane, where `Memb_Pars` is a vector of membrane parameters

```
Memb_Pars = [rou T a]
```

with `rou` = mass density (mass per unit area), `T` = tension, and `a` = radius; and `BC_Spec` is an integer specifying the boundary condition with the options

```
BC_Spec = 0     for a free edge
BC_Spec = 1     for a clamped edge
```

`cirmemb(Memb_Pars, BC_Spec, mn_mode)` computes the eigensolutions of the membrane with `mn_mode = [m n]`, where `m` is the highest number of nodal circles, and `n` is the highest number of nodal diameters. By default, `mn_mode = [4 4]`.

Once `cirmemb` is executed, function `systinfo` can be called at any time to get access to the information on membrane parameters, boundary conditions, and eigensolutions. Also, functions `plotmsh2` and `animode2` can be used to plot and animate a particular mode; see Windows 2.5 and 2.6.

Window 2.5. Functions plotmsh2, plotmsh4

MATLAB Functions: **plotmsh2, plotmsh4**

Purpose:
To plot a mode shape of a circular membrane or plate.

Synopsis:

```
plotmsh2(m, n)
plotmsh2(m, n, IS_Color, grid_pts)
[x, y, w] = plotmsh2(m, n, IS_Color, grid_pts)
```

Description:
`plotmsh2(m, n)` plots the spatial distribution of the displacement of a membrane in mode (m, n), displays the mathematical expression of mode shape function as given in Eq. (16.22), and shows the nodal lines of the mode on which $W_{mn} = 0$.

`plotmsh2(m, n, IS_Color, grid_pts)` plots the mode shape with specified plotting control parameters, where `IS_Color` is an integer specifying the style of the mode shape (color, mesh or surface) according to Table 16.2.1, and `grid_pts = [M N]` with `M, N` being integers specifying an `M`-by-`N` mesh of grid points for the mode shape plot. By default, `IS_Color = 19, M = 30`, and `N = 60`.

`[x, y, w] = plotmsh2(m, n, IS_Color, grid_pts)` returns matrices `x`, `y`, and `w` by which the three-dimensional profile of the mode shape can be generated by the MATLAB function `Mesh(x, y, w)` or `Surf(x, y, w)`. Here matrices `x` and `y` contain the coordinates of the grid points at which the modal displacement is computed, and matrix `w` contains the values of the modal displacement at the grid points.

The utility of function `plotmsh4` is the same as that of `plotmsh2`, except that it is for a circular plate (see Section 16.4).

Window 2.6. Functions animode2, animode4

MATLAB Functions: **animode2, animode4**

Purpose:
To animate a mode of vibration of a circular membrane or plate in its spatial domain.

Synopsis:

```
animode2(m, n)
animode2(m, n, IS_Color, grid_pts, tf, n_frames, IS_control)
F = animode2(m, n, IS_Color, grid_pts, tf, n_frames, IS_control)
```

Description:
`animode2(m, n)` animates the vibration of a circular membrane in mode `(m, n)`.

Window 2.6. Functions animode2, animode4 (*Continued*)

`animode2(m, n, IS_Color, grid_pts, tf, n_frames, IS_control)` animates mode (m, n) with specified control parameters, where `IS_Color` is an integer specifying the style of the modal displacement according to Table 16.2.1, `grid_pts = [M N]` specifying an M-by-N mesh of grid points for animation, `tf` is total animation time, `n_frames` is the number of frames in animation, and `IS_control` is an animation control parameter with the options:

```
IS_control = 0    play frames continuously
IS_control = 1    play frames one by one
```

By default, `IS_Color = 19, M = 30, N = 60, tf` = 8*T, with T being the period of the mode, `n_frames = 97`, and `IS_control = 0`. Each of the above control parameters can be replaced by `[ ]` as a placeholder for its default value.

`F = animode2(m, n, IS_Color, grid_pts, tf, n_frames, IS_control)` returns a set of frames of membrane vibration in a matrix F, which can be played by the MATLAB function `movie(F)`.

The utility of function `animode4` is the same as that of `animode2`, except that it is for a circular plate (see Section 16.4).

EXAMPLE 2.3

For a circular membrane of clamped edge and parameters $\rho = 0.2$, $T = 10$, $a = 0.6$,

```
>> cirmemb([0.2 10 0.6], 1)
```

gives the natural frequencies of the membrane as follows

```
Natural Frequencies:
--------------------
 w_mn       n = 0       1        2         3         4
-------------------------------------------------------
n =
      0   28.3411   45.1571   60.5239   75.1909   89.4295
 1.0000   65.0547   82.6795   99.1982  115.0348  130.3989
 2.0000  101.9852  119.8955  136.9411  153.3856  169.3820
 3.0000  138.9646  157.0212  174.3720  191.1954  207.6062
 4.0000  175.9626  194.1082  211.6585  228.7422  245.4478
```

EXAMPLE 2.4

Consider a circular membrane of free edge, with parameters $\rho = 0.07$, $T = 20$, $a = 1$. The commands

```
>> cirmemb([0.07 20 1], 0)
>> plotmsh2(1,0)
>> plotmsh2(1,1)
```

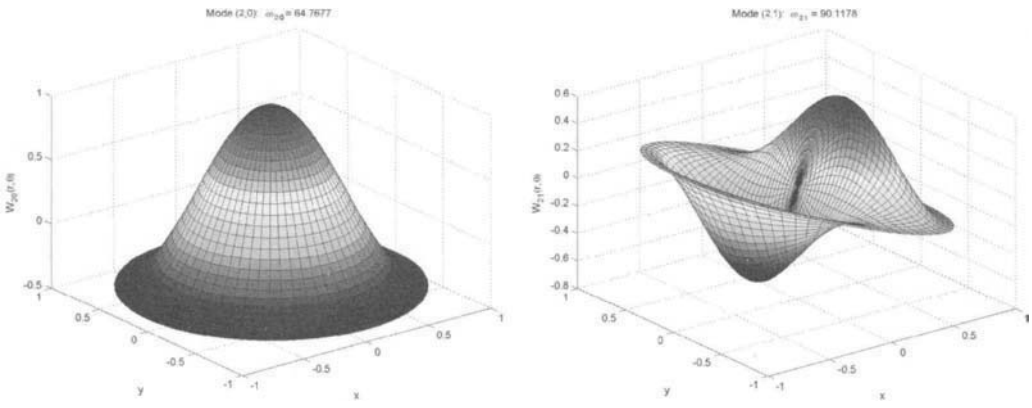


FIGURE 16.2.5 The spatial distribution of modes (1,0) and (1,1).

plot the spatial distributions of modes (1,0) and (1,1) in Fig 16.2.5, and obtain the analytical expressions of the mode shapes as follows

$$W_{10}(r,\theta) = J_1(3.8317r), \qquad 0 \le r \le 1, \quad 0 \le \theta \le 2\pi$$
$$W_{11}(r,\theta) = J_1(5.3314r)\cos\theta, \quad 0 \le r \le 1, \quad 0 \le \theta \le 2\pi$$

where J_1 is Bessel function of the first kind.

16.3 Free Vibration of Rectangular Plates

16.3.1 Plate Theory

Equation of Motion In Fig. 16.3.1, a rectangular plate of length a, width b, and thickness h is made of homogeneous, isotropic elastic material. The transverse displacement (lateral deflection) $w(x,y,t)$ of the neutral surface of the plate in free vibration is governed by the differential equation (Reference 5)

$$D\nabla^4 w(x,y,t) + \rho\frac{\partial^2 w(x,y,t)}{\partial t^2} = 0, \quad 0 \le x \le a, \quad 0 \le y \le b \tag{16.29}$$

where the biharmonic operator

$$\nabla^4 = \frac{\partial^4}{\partial x^4} + 2\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2} + \frac{\partial^4}{\partial y^4} \tag{16.30}$$

ρ is the mass density (mass per unit area), and D is the bending stiffness or flexural rigidity of the plate given by $D = \dfrac{Eh^3}{12(1-\nu^2)}$, with E being Young's modulus and ν Poisson's ratio.

Internal Forces Consider a small rectangular element of the plate with its edges parallel to x- and y-axes, respectively. Acting on these edges are bending moments M_x and M_y, twisting moments M_{xy} and M_{yx}, and shear forces Q_x and Q_y; see Fig. 16.3.2, where the symbol ⟶ indicates the direction of a moment by the right-hand rule. These internal forces are related to

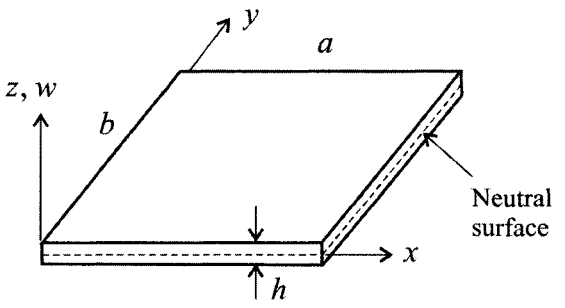


FIGURE 16.3.1 A rectangular plate in transverse vibration.

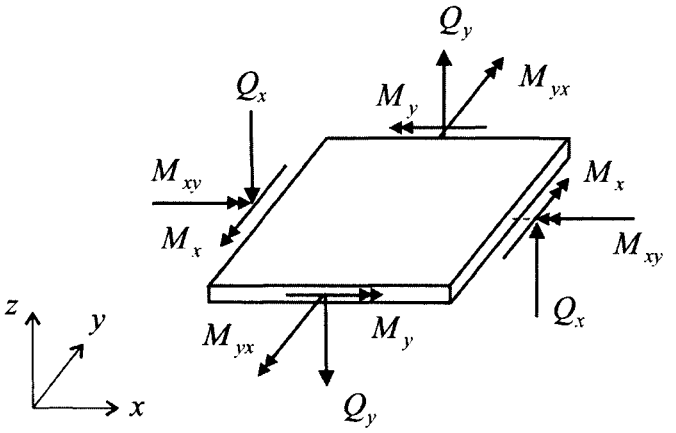


FIGURE 16.3.2 Internal forces on a plate element.

the transverse displacement by

$$
\begin{aligned}
M_x &= -D\left(\frac{\partial^2 w}{\partial x^2} + \nu\frac{\partial^2 w}{\partial y^2}\right), \quad M_y = -D\left(\frac{\partial^2 w}{\partial y^2} + \nu\frac{\partial^2 w}{\partial x^2}\right) \\
M_{xy} &= M_{yx} = -D(1-\nu)\frac{\partial^2 w}{\partial x \partial y} \\
Q_x &= \frac{\partial M_{xy}}{\partial y} + \frac{\partial M_x}{\partial x} = -D\frac{\partial}{\partial x}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right) \\
Q_y &= \frac{\partial M_y}{\partial y} + \frac{\partial M_{xy}}{\partial x} = -D\frac{\partial}{\partial y}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right)
\end{aligned}
\tag{16.31}
$$

Strain, Stress, and Curvature According to Kirchhoff's assumptions, the strains caused by bending are given by

$$
\varepsilon_x = z\kappa_x, \quad \varepsilon_y = z\kappa_y, \quad \gamma_{xy} = 2z\kappa_{xy} \tag{16.32}
$$

where z is the distance from the neutral surface of the plate, and κ_x, κ_y, and κ_{xy} are the curvatures of the neutral surface shown in Fig. 16.3.1. The curvatures are related to the transverse displacement by

$$\kappa_x = -\frac{\partial^2 w}{\partial x^2}, \quad \kappa_y = -\frac{\partial^2 w}{\partial y^2}, \quad \kappa_{xy} = -\frac{\partial^2 w}{\partial x \partial y} \tag{16.33}$$

For linear elastic material, the stress components of the plate are given by

$$\sigma_x = \frac{Ez}{1-\nu^2}\left(\kappa_x + \nu\kappa_y\right), \quad \sigma_y = \frac{Ez}{1-\nu^2}\left(\kappa_y + \nu\kappa_x\right), \quad \tau_{xy} = \frac{Ez}{1+\nu}\kappa_{xy} \tag{16.34}$$

Moment-Curvature Relation The moments of the plate are defined by

$$M_x = \int_{-h/2}^{h/2} \sigma_x z dz, \quad M_y = \int_{-h/2}^{h/2} \sigma_y z dz, \quad M_{xy} = \int_{-h/2}^{h/2} \tau_{xy} z dz$$

which, by Eqs. (16.34), leads to moment-curvature relation

$$\begin{pmatrix} M_x \\ M_y \\ M_{xy} \end{pmatrix} = D \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \begin{pmatrix} \kappa_x \\ \kappa_y \\ 2\kappa_{xy} \end{pmatrix} \tag{16.35}$$

With Eq. (16.35), the stresses of the plate can be expressed in terms of moments, i.e.,

$$\sigma_x = \frac{12M_x}{h^3}z, \quad \sigma_y = \frac{12M_y}{h^3}z, \quad \tau_{xy} = \frac{12M_{xy}}{h^3}z \tag{16.36}$$

Also, substitution of Eq. (16.33) into Eq. (16.35) yields Eq. (16.31).

Boundary Conditions Three classical boundaries of the plate are described as follows.

Simply Supported Edges At edge $x = 0$ or $x = a$

$$\begin{aligned} w &= 0 \\ \frac{\partial^2 w}{\partial x^2} &= 0 \end{aligned} \tag{16.37a}$$

and at edge $y = 0$ or $y = b$

$$\begin{aligned} w &= 0 \\ \frac{\partial^2 w}{\partial y^2} &= 0 \end{aligned} \tag{16.37b}$$

Clamped Edges At edge $x = 0$ or $x = a$

$$\begin{aligned} w &= 0 \\ \frac{\partial w}{\partial x} &= 0 \end{aligned} \tag{16.38a}$$

and at edge $y = 0$ or $y = b$

$$\begin{aligned} w &= 0 \\ \frac{\partial w}{\partial y} &= 0 \end{aligned} \tag{16.38b}$$

Free Edges At edge at $x = 0$ or $x = a$

$$\begin{aligned} M_x &= -D\left(\frac{\partial^2 w}{\partial x^2} + \nu\frac{\partial^2 w}{\partial y^2}\right) = 0 \\ V_x &= Q_x + \frac{\partial M_{xy}}{\partial y} = -D\left[\frac{\partial^3 w}{\partial x^3} + (2-\nu)\frac{\partial^3 w}{\partial x \partial y^2}\right] = 0 \end{aligned} \tag{16.39a}$$

and at edge $y = 0$ or $y = b$

$$\begin{aligned} M_y &= -D\left(\frac{\partial^2 w}{\partial y^2} + \nu\frac{\partial^2 w}{\partial x^2}\right) = 0 \\ V_y &= Q_y + \frac{\partial M_{xy}}{\partial x} = -D\left[\frac{\partial^3 w}{\partial y^3} + (2-\nu)\frac{\partial^3 w}{\partial x^2 \partial y}\right] = 0 \end{aligned} \tag{16.39b}$$

where V_x and V_y are effective transverse forces per unit length.

Kinetic and Strain Energy The kinetic energy of the plate is

$$T = \frac{1}{2}\iint_A \rho\left(\frac{\partial w}{\partial t}\right)^2 dxdy \tag{16.40}$$

The strain energy of the plate is given by

$$U = \frac{1}{2}\int_V \left(\sigma_x\varepsilon_x + \sigma_y\varepsilon_y + \tau_{xy}\gamma_{xy}\right) dV = \frac{1}{2}\iint_A \left(M_x\kappa_x + M_y\kappa_y + 2M_{xy}\kappa_{xy}\right) dxdy$$

which, by Eqs. (16.31) and (16.33), leads to

$$U = \frac{1}{2}\iint_A D\left[\left(\frac{\partial^2 w}{\partial x^2}\right)^2 + \left(\frac{\partial^2 w}{\partial y^2}\right)^2 + 2\nu\frac{\partial^2 w}{\partial x^2}\frac{\partial^2 w}{\partial Y^2} + 2(1-\nu)\left(\frac{\partial^2 w}{\partial X \partial y}\right)^2\right] dxdy \tag{16.41}$$

The above energy functionals are useful in development of approximate solution methods for the plate vibration problem, such as Rayleigh-Ritz method and finite element methods.

16.3.2 Eigenvalue Problem

A fundamental issue in free vibration analysis of plates is to solve the differential equation (16.29) subject to appropriate boundary conditions. By separation of variables, the transverse displacement of a plate can be expressed as

$$w(x,y,t) = W(x,y)\cos(\omega t + \theta) \tag{16.42}$$

where θ is an arbitrary constant, $W(x,y)$ is an unknown function, and ω is an unknown parameter to be determined. Substituting Eq. (16.42) into Eq. (16.29) yields

$$\left(\frac{\partial^4}{\partial x^4} + 2\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2} + \frac{\partial^4}{\partial y^4}\right) W(x,y) - \frac{\rho\omega^2}{D} W(x,y) = 0 \tag{16.43}$$

Equation (16.43), along with appropriate boundary conditions, defines an eigenvalue problem of the plate, in which ω is an eigenvalue and $W(x,y)$ is an eigenfunction. Physically, ω is a natural frequency of the plate while $W(x,y)$ is the associate mode shape. The boundary conditions for Eq. (16.43) are given by Eqs. (16.37) to (16.39) with $w(x,y,t)$ replaced by $W(x,y)$.

The natural frequencies of a plate can be expressed by $\omega = \chi\sqrt{D/\rho}$, where the parameter χ depends on the material and geometric properties of the plate. If the plate does not have a free edge, χ is a function of the length a and width b only, and does not depend on Poisson's ratio ν. This is due to the fact that the boundary conditions (16.37) and (16.38) do not involve ν. On the other hand, if the plate has at least one free edge, Eqs. (16.39) imply that χ is dependent upon the plate length and width, as well as Poisson's ratio.

In what follows, three solution techniques for the above eigenvalue problem are introduced: Navier solution method, Lévy solution method, and a finite element method. The Navier and Lévy methods are analytical methods, while the finite element method is a numerical one.

Navier Solutions for Simply Supported Plates For a plate that is simply supported along all edges, assume its displacement as

$$W(x,y) = A\sin\frac{m\pi x}{a}\sin\frac{n\pi y}{b} \tag{16.44}$$

where m and n are integers, and A is a nonzero constant. The expression in Eq. (16.44) automatically satisfies the boundary conditions specified in Eqs. (16.37), and when substituted into Eq. (16.43), yields the following characteristic equation

$$\pi^4\left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right)^2 - \frac{\rho\omega^2}{D} = 0 \tag{16.45}$$

The roots of Eq. (16.45) are the natural frequencies of the plate, namely

$$\omega_{mn} = \pi^2\left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right)\sqrt{\frac{D}{\rho}}, \quad m,n = 1,2,\ldots \tag{16.46}$$

The mode shapes are given by Eq. (16.44), with m and n representing the number of half sine waves in the x and y directions, respectively. The above process is known as Navier solution.

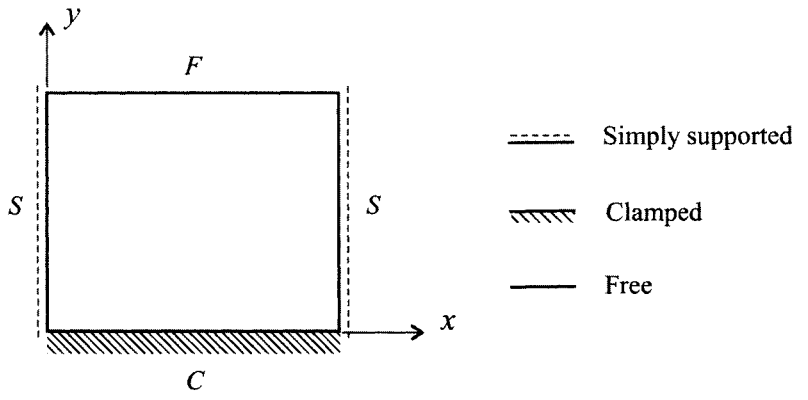


FIGURE 16.3.3 Illustration of plate boundary conditions by letters S, C, and F.

Lévy Solutions for Plates with Two Opposite Edges Simply Supported The plate in consideration is simply supported at edges $x = 0$ and $x = a$, and has arbitrary support (boundary) conditions at the other two edges $y = 0$ and $y = b$; see Fig. 16.3.3 for example, where letters S, C, and F denote simply supported, clamped, and free edges, respectively. The eigenfunction of such a plate is assumed as

$$W(x, y) = Y(y) \sin \frac{m\pi x}{a} \tag{16.47}$$

where m is an integer and $Y(y)$ is an unknown function to be determined. The sine function in the previous equation automatically satisfies the boundary conditions at $x = 0$ and b. Equation (16.47), which is known as Lévy-type solution, is substituted into Eq. (16.43), to obtain

$$\frac{d^4}{dy^4} Y(y) - 2\alpha_m^2 \frac{d^2}{dy^2} Y(y) + \left(\alpha_m^4 - \gamma^4\right) Y(y) = 0 \tag{16.48}$$

with

$$\alpha_m = \frac{m\pi}{a}, \quad \gamma^4 = \frac{\rho\omega^2}{D} \tag{16.49}$$

Also, the boundary conditions for Y are derived as follows:

Simply supported at $y = 0$ or b:	$Y = 0, \quad Y'' = 0$	(16.50a)
Clamped at $y = 0$ or b:	$Y = 0, \quad Y' = 0$	(16.50b)
Free at $y = 0$ or b:	$Y'' - \nu\alpha_m^2 Y = 0, \quad Y''' - (2 - \nu)\alpha_m^2 Y'' = 0$	(16.50c)

where $Y' = dY/dy$.

The solution of the differential equation (16.48) is in one of the three forms:

(i) For $0 < \gamma < \alpha_m$

$$Y(y) = A \cosh \alpha y + B \sinh \alpha y + C \cosh \beta y + D \sinh \beta y \tag{16.51}$$

where $\alpha = \sqrt{\alpha_m^2 + \gamma^2}$ and $\beta = \sqrt{\alpha_m^2 - \gamma^2}$

(ii) For $\gamma = \alpha_m$

$$Y(y) = A\cosh\alpha y + B\sinh\alpha y + Cy + D \tag{16.52}$$

where $\alpha = \sqrt{2}\,\alpha_m$.

(iii) For $\gamma > \alpha_m$

$$Y(y) = A\cosh\alpha y + B\sinh\alpha y + C\cos\beta y + D\sin\beta y \tag{16.53}$$

where $\alpha = \sqrt{\alpha_m^2 + \gamma^2}$ and $\beta = \sqrt{\gamma^2 - \alpha_m^2}$.

The characteristic equation of the plate is obtained by substituting Eqs. (16.51) to (16.53) into the boundary conditions (16.50), from which the natural frequencies of the plate can be computed. As an example, for the plate shown in Fig. 16.3.3, the boundary conditions at edges $y = 0$ and b are

$$Y(0) = 0, \quad Y'(0) = 0$$
$$Y''(b) - \nu\alpha_m^2 Y(b) = 0, \quad Y'''(b) - (2-\nu)\alpha_m^2 Y'(b) = 0.$$

For $\gamma > \alpha_m$, substituting Eq. (16.53) into the boundary conditions gives

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ d_1\cosh\alpha b & d_1\sinh\alpha b & -d_2\cos\beta b & -d_2\sin\beta b \\ d_3\sinh\alpha b & d_3\cosh\alpha b & d_4\sin\beta b & -d_4\cos\beta b \end{bmatrix} \begin{Bmatrix} A \\ B \\ C \\ D \end{Bmatrix} = 0 \tag{16.54}$$

where $d_1 = \alpha^2 - \nu\alpha_m^2$, $d_2 = \beta^2 + \nu\alpha_m^2$, $d_3 = \alpha^3 - (2-\nu)\alpha_m^2\alpha$, and $d_4 = \beta^3 + (2-\nu)\alpha_m^2\beta$. Vanishing the determinant of the matrix in Eq. (16.54) yields the characteristic equation of the plate

$$d_1 d_3 - d_2 d_4 \cos 2\beta b + (d_1 d_4 - d_2 d_3)(\cosh\alpha b \cdot \cos\beta b + \sinh\alpha b \cdot \sin\beta b) = 0. \tag{16.55}$$

Solution of the above equation gives the natural frequencies of the plate. Also, with a known natural frequency, a nontrivial solution of coefficients A, B, C, and D from Eq. (16.54) gives the associate mode shape by Eqs. (16.47) and (16.53). The above solution procedure is also applicable to the cases of $0 < \gamma < \alpha_m$ and $\gamma = \alpha_m$.

A Finite Element Method for General Boundary Conditions Free vibration analysis of plates with arbitrary boundary conditions generally has to depend on numerical methods, such as finite element methods, Rayleigh-Ritz methods, and finite difference methods. In this section, the eigensolutions of rectangular plates are obtained by a finite element method. The reader may refer to References 6 to 10 for fundamental issues on finite element methods.

In Fig. 16.3.4, a rectangular plate of length L and width H is divided into an M-by-N mesh of rectangular elements. Consider a typical element (e) in Fig. 16.3.5, which is of length $2a$, width $2b$, and thickness h, and has four nodes i, j, l, and m. Define the local coordinates

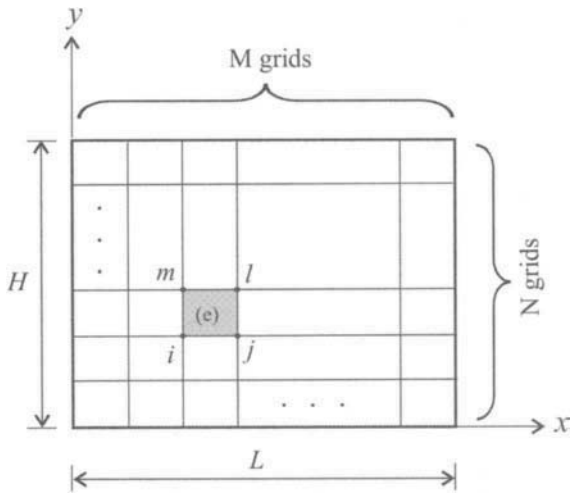


FIGURE 16.3.4 A mesh of rectangular elements.

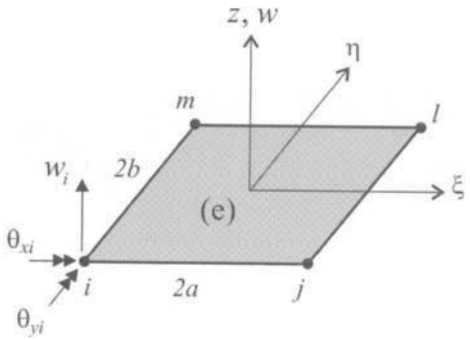


FIGURE 16.3.5

$\xi = x/a$ and $\eta = y/b$ that are originated at the center of the element. The transverse displacement of the element is interpolated as

$$w = \left[\mathbf{N}_i(\xi,\eta) \quad \mathbf{N}_j(\xi,\eta) \quad \mathbf{N}_l(\xi,\eta) \quad \mathbf{N}_m(\xi,\eta)\right]\{u\}_e = [\mathbf{N}(\xi,\eta)]\{u\}_e \tag{16.56}$$

where

$$\{u\}_e = \begin{pmatrix} w_i & \theta_{xi} & \theta_{yi} & \cdots & w_m & \theta_{xm} & \theta_{ym} \end{pmatrix}^T \tag{16.57}$$

with w_k being the transverse displacement at node k, θ_{xk} the rotation about the x axis, and θ_{yk} the rotation about the y axis, and

$$\mathbf{N}_k^T(\xi,\eta) = \frac{1}{8}\begin{bmatrix} (1+\xi_k\xi)(1+\eta_k\eta)(2+\xi_k\xi+\eta_k\eta-\xi^2-\eta^2) \\ b(1+\xi_k\xi)(\eta_k+\eta)(\eta^2-1) \\ -a(\xi_k+\xi)(\xi^2-1)(1+\eta_k\eta) \end{bmatrix}, \quad k = i,j,l,m \tag{16.58}$$

with ξ_k and η_k being the coordinates of node k (i.e., $\xi_k = \pm 1$ and $\eta_k = \pm 1$). The elements of $[\mathbf{N}(\xi, \eta)]$ are referred to as *shape functions*. The rotations are related to the displacement by

$$\theta_x = \frac{\partial w}{\partial y}, \quad \theta_y = -\frac{\partial w}{\partial x} \tag{16.59}$$

With each node having three degrees of freedom (w_k, θ_{xk}, θ_{yk}), the element has 12 degrees of freedom. In the literature, this element is known as ACM element; see Reference 10 for instance.

Substitute Eq. (16.56) into to Eqs. (16.40) and (16.41), to obtain the kinetic energy and strain energy of the element

$$\begin{aligned} T_e &= \frac{1}{2} \{\dot{u}\}_e^T [m]_e \{\dot{u}\}_e \\ U_e &= \frac{1}{2} \{u\}_e^T [k]_e \{u\}_e \end{aligned} \tag{16.60}$$

where the element mass matrix is given by

$$[m]_e = \rho ab \int_{-1}^{1} \int_{-1}^{1} [\mathbf{N}(\xi, \eta)]^T [\mathbf{N}(\xi, \eta)] d\xi d\eta \tag{16.61}$$

and the element stiffness matrix is given by

$$[k]_e = Dab \int_{-1}^{1} \int_{-1}^{1} [\mathbf{B}]^T [\mathbf{C}] [\mathbf{B}] d\xi d\eta \tag{16.62}$$

with D being the bending stiffness of the plate and

$$[\mathbf{B}] = \begin{bmatrix} \frac{1}{a^2} \frac{\partial^2}{\partial \xi^2} \\ \frac{1}{b^2} \frac{\partial^2}{\partial \eta^2} \\ \frac{2}{ab} \frac{\partial^2}{\partial \xi \partial \eta} \end{bmatrix} [\mathbf{N}(\xi, \eta)], \quad [\mathbf{C}] = \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix} \tag{16.63}$$

The above mass and stiffness matrices can be easily computed via numerical integration. The explicit expressions of these matrices are also available; see References 8 and 10.

For the entire plate with Ne finite elements ($Ne = M \times N$), the total kinetic energy and strain energy are

$$\begin{aligned} T &= \sum_{e=1}^{Ne} \frac{1}{2} \{\dot{u}\}_e^T [m]_e \{\dot{u}\}_e = \frac{1}{2} \{\dot{q}\}^T [M] \{\dot{q}\} \\ U_e &= \sum_{e=1}^{Ne} \frac{1}{2} \{u\}_e^T [k]_e \{u\}_e = \frac{1}{2} \{q\}^T [K] \{q\} \end{aligned} \tag{16.64}$$

where $\{q\}$ is a vector consisting of all the independent nodal displacements (w_k, θ_{xk}, θ_{yk}), and $[M]$ and $[K]$ are the global mass and stiffness matrices that are assembled from the element

matrices $[m]_e$ and $[k]_e$, respectively. Here, $[M]$ and $[K]$ are formed after the boundary conditions of the plate are introduced.

With (16.64), Lagrange's equation

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \{\dot{q}\}}\right) + \frac{\partial U}{\partial \{q\}} = 0$$

leads to the second-order differential equation

$$[M]\{\ddot{q}\} + [K]\{q\} = 0 \tag{16.65}$$

which represents a discretized model of the plate in free vibration. By assuming

$$\{q\} = \{V\} \sin \omega t \tag{16.66}$$

Eq. (16.65) is reduced to the eigenvalue problem

$$\omega^2 [M]\{V\} = [K]\{V\} \tag{16.67}$$

where the parameter ω is a natural frequency of the plate and the vector $\{V\}$ describes the associate mode shape. The eigenvalues are the roots of the characteristic equation

$$\det\left(-\omega^2 [M] + [K]\right) = 0 \tag{16.68}$$

Solution methods for the above eigenvalue problem are well developed.

In summary, free vibration analysis of a plate by a finite element method takes three steps:

(a) Compute the element mass and stiffness matrices $[m]_e$, $[k]_e$;
(b) Assemble the global mass and stiffness matrices $[M]$, $[K]$; and
(c) Solve Eq. (16.67) for the natural frequencies and mode shapes of the plate.

16.3.3 Solution by the Toolbox

The Toolbox of this chapter provides a set of MATLAB functions for computation and illustration of eigensolutions of rectangular plates in free vibration. Both the analytical and numerical solutions presented in the previous section can be obtained by these functions.

Navier and Lévy Solutions In Fig. 16.3.6, a plate of length a and width b is simply supported at the opposite edges $x = 0$ and $x = a$, and has arbitrary boundary conditions at the other edges $y = 0$ and $y = b$. For such a plate, function `recplateLV` yields eigensolutions of Navier or Lévy type; see Window 3.1.

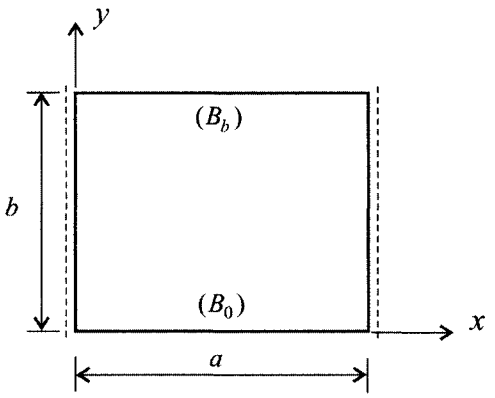


FIGURE 16.3.6 A plate simply-supported at two opposite edges.

Window 3.1. Function recplateLV

MATLAB Function: **recplateLV**

Purpose:
To obtain Navier or Lévy solutions of the natural frequencies and mode shapes of a rectangular plate with at least two opposite edges simply supported, as shown in Fig. 16.3.6.

Synopsis:

```
recplateLV(Plate_Pars, BC_Spec)
recplateLV(Plate_Pars, BC_Spec, mn_mode)
```

Description:
`recplateLV(Plate_Pars, BC_Spec)` computes the natural frequencies and mode shapes of the plate. Here `Plate_Pars` is a vector of plate parameters

```
Plate_Pars = [rou E mu a b h]
```

with `rou` = mass density (mass per unit area), `E` = Young's modulus, `mu` = Poisson's ratio, `a` = length, `b` = height, and `h` = thickness; and `BC_Spec` is a vector of the form

```
BC_Spec = [B0 Bb]
```

where `B0` and `Bb` are integers specifying the boundary conditions at edges $y = 0$ and $y = b$, respectively, with the following options:

```
B0 or Bb = 0   for a free edge
B0 or Bb = 1   for a simply supported edge
B0 or Bb = 2   for a clamped edge
```

For instance, `BC_Spec = [0 2]` assigns a free edge at $y = 0$, and a clamped edge at $y = b$.

Window 3.1. Function recplateLV (*Continued*)

recplateLV(Plate_Pars, BC_Spec, mn_mode) computes the natural frequencies and mode shapes of the plate with a specified vector mn_mode = [m n], where m is the highest wave number of the mode shapes in the x direction, and n is the highest wave number in the y direction. By default, mn_mode = [4 4].

Note. After recplateLV is executed, function systinfo can be called at any time, such as

```
>> systinfo
```

to get access to the information on plate parameters, boundary conditions, and natural frequencies.

EXAMPLE 3.1

Consider a plate with simply supported edges at $x = 0$ and a, a free edge at $y = 0$, and a clamped edge at $y = b$. Let the plate parameters be

$$\rho = 0.02, \quad E = 1.5 \times 10^4, \quad \nu = 0.32$$
$$a = 3, \quad b = 2, \quad h = 0.01$$

The MATLAB commands

```
>> Plate_Pars = [0.02 1.5e4 0.32 3 2 0.01];
>> BC_Spec = [0 2];
>> recplateLV(Plate_Pars, BC_Spec)
```

yield the natural frequencies of the plate as follows

```
Natural Frequencies:
--------------------
 w_mn       n = 1      2          3         4
-----------------------------------------------
m =
 1.0000    0.4894    1.7851    4.3836    8.2834
 2.0000    1.3223    2.6973    5.3037    9.1978
 3.0000    2.7431    4.1452    6.7916   10.6982
 4.0000    4.7466    6.1505    8.8314   12.7651

Here  w_mn = natural frequency of wave numbers m & n
        m = wave number in x direction
        n = wave number in y direction
```

Besides recplateLV, function plotmsh3 from the Toolbox can be used to plot the mode shapes of the plate, and function animode3 can be used to animate the vibration of the plate in a particular mode; see Windows 2.2 and 2.3.

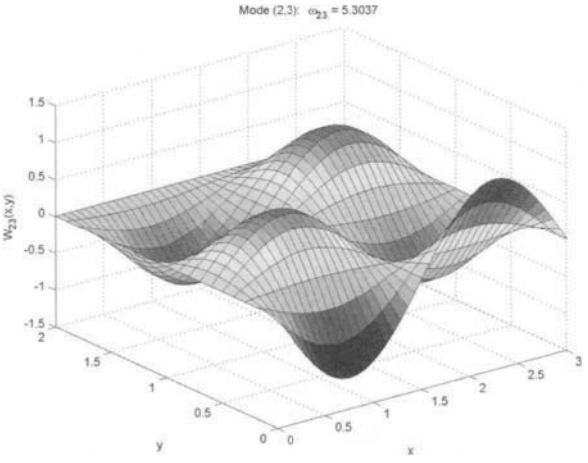


FIGURE 16.3.7 The plate displacement distribution in mode (2,3).

EXAMPLE 3.2

Consider the plate in Example 3.1.

```
>> recplateLV([0.02 1.5e4 0.32 3 2 0.01], [0 2])
>> plotmsh3(2,3)
```

plots the spatial profile of mode (2,3) in Figs. 16.3.7 and 16.3.8, and displays the mathematical expression of the mode shape as follows

$$W(x, y) = \sin(2\pi x/3)\, Y(y)$$

with

$$Y(y) = -0.47065e^{-4.9483(2-y)} - 0.46022e^{-4.9483y} - 0.62065\cos(3.9639y) + 0.426\sin(3.9639y).$$

Finite Element Solutions For a plate with general boundary conditions, function `recplateFE` is available for computing the eigensolutions of the plate via the finite element method presented in Section 16.3.2; see Window 3.2.

Window 3.2. Function `recplateFE`

MATLAB Function: **`recplateFE`**

Purpose:
To compute the eigensolutions of a rectangular plate with general boundary conditions via the finite element method described in Section 16.3.2.

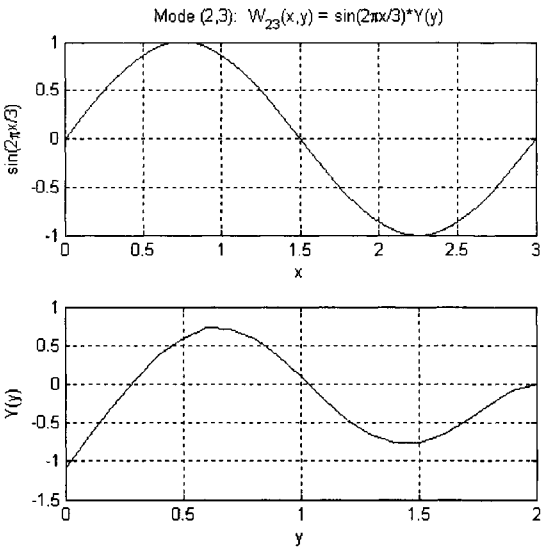


FIGURE 16.3.8 The distribution of W_{23} in the x and y directions.

Window 3.2. Function recplateFE (*Continued*)

Synopsis:

```
recplateFE(Plate_Pars, BC_Spec)
recplateFE(Plate_Pars, BC_Spec, n_modes, FE_mesh)
```

Description:

`recplateFE(Plate_Pars, BC_Spec)` computes the natural frequencies and mode shapes of the plate. Here `Plate_Pars` is a vector of plate parameters

```
Plate_Pars = [rou E mu a b h]
```

with `rou` = mass density (mass per unit area), `E` = Young's modulus, `mu` = Poisson's ratio, `a` = length, `b` = height, and `h` = thickness; `BC_Spec` is a boundary specification vector of the form

```
BC_Spec = [J1 J2 J3 J4]
```

in which `Jk` is an integer specifying the boundary conditions at the edge (`Bk`) of the plate shown in Fig. 16.3.9, with the options

```
Jk = 0   for a free edge
Jk = 1   for a simply supported edge
Jk = 2   for a clamped edge
```

`recplateFE(Plate_Pars, BC_Spec, n_modes, FE_mesh)` computes the eigensolutions of the plate with specified control parameters, where `n_modes` is the highest mode number involved in computation, and `FE_mesh = [M N]`, with integers `M`, `N` specifying an M-by-N mesh of elements. By default, `n_modes` = 8, and `FE_mesh = [10 10]`.

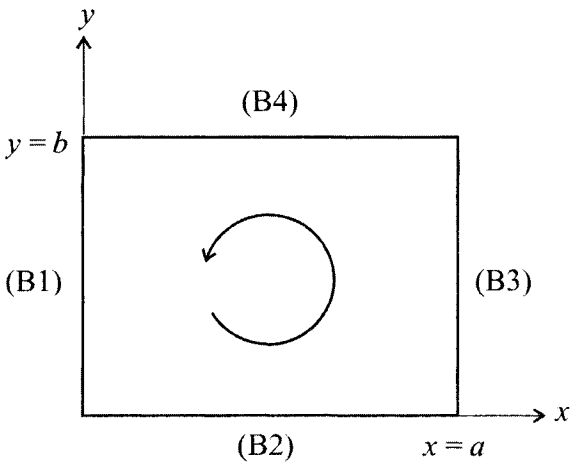


FIGURE 16.3.9 The labeled boundaries of a rectangular plate.

After `recplateFE` is executed, function `systinfo` can be called at any time to get access to the information on plate parameters, boundary condidions, and natural frequencies. Also, functions `plotmsh3f` and `animode3f` can be used to plot and animate a particular mode of vibration of the plate; see Windows 3.3 and 3.4.

Window 3.3. Function `plotmsh3f`

MATLAB Function: **`plotmsh3f`**

Purpose:
To plot a mode shape of a rectangular plate with arbitrary boundary conditions, based on the finite element simulation by function `recplateFE`.

Synopsis:

```
plotmsh3f(n)
plotmsh3f(n, IS_Color, grid_pts)
[x, y, w] = plotmsh3f(n, IS_Color, grid_pts)
```

Description:
`plotmsh3f(n)` plots the displacement of a plate in the nth mode and shows the nodal lines of the mode shape on which the plate displacement is zero.

`plotmsh3f(n, IS_Color, grid_pts)` plots the nth mode shape with specified control parameters, where `IS_Color` is an integer specifying the style of the mode shape according to Table 16.2.1, and `grid_pts = [p q]` with `p`, `q` being positive integers specifying a `pM`-by-`qN` mesh of grid points in the plate domain for plotting the mode shape. Here, `M` and `N` are the finite element mesh numbers assigned by function `recplateFE`. By default, `IS_Color = 19`, and `grid_pts = [4 4]`. Each of `IS_Color` and `grid_pts` can be replaced by `[ ]` as a placeholder for its default value.

`[x, y, w] = plotmsh3f(n, IS_Color, grid_pts)` returns matrices `x`, `y`, and `w` by which the three-dimensional profile of the mode shape can be generated by the MATLAB function `mesh(x, y, w)` or `surf(x, y, w)`.

Window 3.4. Function animode3f

MATLAB Function: **animode3f**

Purpose:
To animate a mode of vibration of a rectangular plate based on the finite element simulation by function recplateFE.

Synopsis:

```
animode3f(n)
animode3f(n, IS_Color, grid_pts, tf, n_frames, IS_control)
F = animode3f(n, IS_Color, grid_pts, tf, n_frames, IS_control)
```

Description:
animode3f(n) animates the nth mode of vibration of the plate.

animode3f(n, IS_Color, grid_pts, tf, n_frames, IS_control) animates the nth mode of vibration with specified control parameters, where IS_Color and grid_pts are the same as those for function plotmsh3f (Window 3.3), tf is total animation time, n_frames is the number of frames in the animation, and IS_control is a parameter controlling the animation with the following two options:

IS_control = 0	play frames continuously
IS_control = 1	play frames one by one

By default, IS_Color = 19, grid_pts = [4 4], tf = 8*T with T being the period of the mode, n_frames = 97, and IS_control = 0. Each of the above parameters can be replaced by [ ] as a placeholder for its default value.

F = animode3f(n, IS_Color, grid_pts, tf, n_frames, IS_control) returns a set of frames of plate vibration in a matrix F, which can be played by the MATLAB function movie(F).

EXAMPLE 3.3

Consider a fully clamped rectangular plate with parameters

$$\rho = 0.015, \quad E = 7 \times 10^3, \quad \nu = 0.3$$

$$a = 4, \quad b = 2, \quad h = 0.02$$

The command

```
>> recplateFE([0.015 7000 0.3 4 2 0.02], [2 2 2 2])
```

yields the natural frequencies of the plate as follows

```
First 8 Natural Frequencies:
----------------------------
wn, natural frequencies in rad/s
fn = wn/(2*pi), natural frequencies in Hertz

    wn        fn
 3.5593    0.5665
 4.5441    0.7232
 6.3611    1.0124
 9.0188    1.4354
 9.2698    1.4753
10.0798    1.6042
11.5627    1.8403
12.4960    1.9888
```

Also,

```
>> plotmsh3f(3)
```

plots the spatial distribution of the third mode shape of the plate in Fig. 16.3.10.

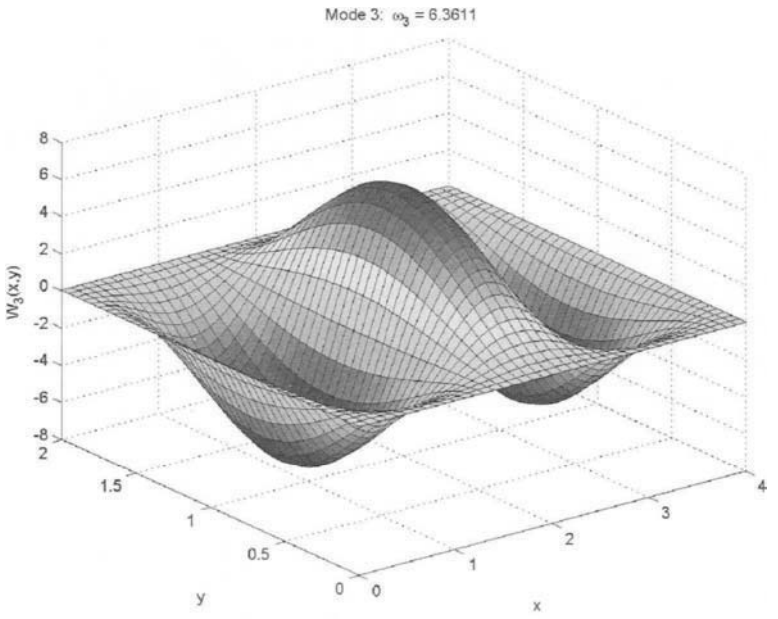


FIGURE 16.3.10

16.4 Free Vibration of Circular Plates

16.4.1 Equations in Polar Coordinates

Equation of Motion For a circular plate made of isotropic elastic material, its vibration can be conveniently described in the polar coordinates r, θ; see Fig. 16.4.1. Let the transverse

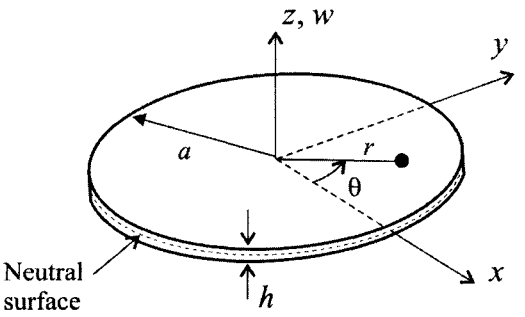


FIGURE 16.4.1 A circular plate in transverse vibration.

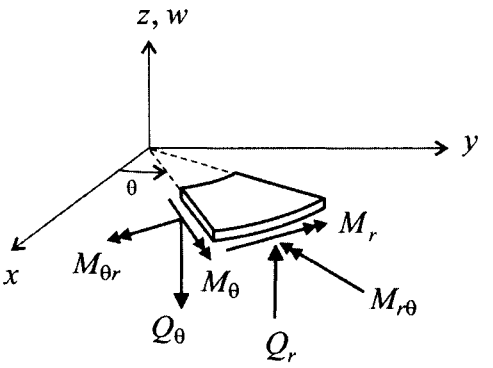


FIGURE 16.4.2 Internal forces on a plate element.

displacement of the plate be written as $w(r, \theta, t)$. The governing equation of motion of the plate in free vibration is

$$D\nabla^4 w(r,\theta,t) + \rho \frac{\partial^2 w(r,\theta,t)}{\partial t^2} = 0, \quad 0 \le r \le a, \quad 0 \le \theta \le 2\pi \tag{16.69}$$

where the biharmonic operator $\nabla^4 = \nabla^2\nabla^2$, with

$$\nabla^2 = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2}; \tag{16.70}$$

ρ is the mass density (mass per unit area); $D = \frac{Eh^3}{12(1-\nu^2)}$ is the bending stiffness or flexural rigidity of the plate, with E being Young's modulus and ν Poisson's ratio; and a and h are the radius and thickness of the plate, respectively.

Internal Forces On a small plate element in Fig. 16.4.2 are acting bending moments M_r and M_θ, twisting moments $M_{r\theta}$ and $M_{\theta r}$, and shear forces Q_r and Q_θ, where the direction of a moment, denoted by the symbol ⟶⟶, is determined by the right-hand rule. The moments and

shear forces are related to the transverse displacement by

$$\begin{aligned} M_r &= -D\left[\frac{\partial^2 w}{\partial r^2} + \nu\left(\frac{1}{r}\frac{\partial w}{\partial r} + \frac{1}{r^2}\frac{\partial^2 w}{\partial \theta^2}\right)\right] \\ M_\theta &= -D\left[\left(\frac{1}{r}\frac{\partial w}{\partial r} + \frac{1}{r^2}\frac{\partial^2 w}{\partial \theta^2}\right) + \nu\frac{\partial^2 w}{\partial r^2}\right] \\ M_{r\theta} &= M_{\theta r} = -D(1-\nu)\left(\frac{1}{r}\frac{\partial^2 w}{\partial r \partial \theta} - \frac{1}{r^2}\frac{\partial w}{\partial \theta}\right) \\ Q_r &= -D\frac{\partial}{\partial r}\nabla^2 w, \quad Q_\theta = -D\frac{1}{r}\frac{\partial}{\partial \theta}\nabla^2 w \end{aligned} \tag{16.71}$$

The moments in Eq. (16.71) can also be rewritten as

$$\begin{pmatrix} M_r \\ M_\theta \\ M_{r\theta} \end{pmatrix} = D\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix}\begin{pmatrix} \kappa_r \\ \kappa_\theta \\ 2\kappa_{r\theta} \end{pmatrix} \tag{16.72}$$

where κ_r et al. are the curvatures of the neutral surface of the plate, and are given by

$$\kappa_r = -\frac{\partial^2 w}{\partial r^2}, \quad \kappa_\theta = -\left(\frac{1}{r}\frac{\partial w}{\partial r} + \frac{1}{r^2}\frac{\partial^2 w}{\partial \theta^2}\right), \quad \kappa_{r\theta} = -\left(\frac{1}{r}\frac{\partial^2 w}{\partial r \partial \theta} - \frac{1}{r^2}\frac{\partial w}{\partial \theta}\right) \tag{16.73}$$

Boundary Conditions The boundary conditions of the plate are specified at the edge $r = a$. Three types of boundary conditions are given below.

Simply Supported Edge

$$(w)_{r=a} = 0, \quad \left(\frac{\partial^2 w}{\partial r^2} + \nu\frac{1}{r}\frac{\partial w}{\partial r}\right)_{r=a} = 0 \tag{16.74a}$$

Clamped Edge

$$(w)_{r=a} = 0, \quad \left(\frac{\partial w}{\partial r}\right)_{r=a} = 0 \tag{16.74b}$$

Free Edge

$$(M_r)_{r=a} = 0, \quad (V_r)_{r=a} = \left(Q_r + \frac{1}{r}\frac{\partial M_{r\theta}}{\partial \theta}\right)_{r=a} = 0 \tag{16.74c}$$

where V_r is an effective transverse force along the edge.

Kinetic and Strain Energy The kinetic energy of the circular plate is

$$T = \frac{1}{2}\iint_A \rho\left(\frac{\partial w}{\partial t}\right)^2 r dr d\theta \tag{16.75}$$

The strain energy of the plate is given by

$$U = \frac{1}{2}\iint_A (M_r\kappa_r + M_\theta\kappa_\theta + 2M_{r\theta}\kappa_{r\theta})\,dxdy$$

$$= \frac{1}{2}\iint_A D\left\{\left(\nabla^2 w\right)^2 - 2(1-\nu)\left[\frac{\partial^2 w}{\partial r^2}\left(\frac{1}{r}\frac{\partial w}{\partial r} + \frac{1}{r^2}\frac{\partial^2 w}{\partial \theta^2}\right) - \left(\frac{1}{r}\frac{\partial^2 w}{\partial r \partial \theta} - \frac{1}{r^2}\frac{\partial w}{\partial \theta}\right)\right]\right\} r dr d\theta \tag{16.76}$$

16.4.2 Modes of Vibration

Assume that the solution of Eq. (16.69) has the form

$$w(r,\theta,t) = W(r,\theta)\cos(\omega t + \alpha) \tag{16.77}$$

where ω and $W(r,\theta)$ are the natural frequency and associate mode shape of the plate, respectively, and α is an arbitrary constant. Substitute Eq. (16.77) into Eq. (16.69) to obtain

$$\left(\nabla^2 + \gamma^2\right)\left(\nabla^2 - \gamma^2\right) W(r,\theta) = 0 \tag{16.78}$$

where γ is a frequency parameter defined by

$$\gamma^4 = \frac{\rho\omega^2}{D}. \tag{16.79}$$

Letting the mode shape function be of the form

$$W(r,\theta) = F(r)\cos n\theta, \quad n = 0,1,2,\ldots \tag{16.80}$$

reduces Eq. (16.78) to

$$\frac{d^2F}{dr^2} + \frac{1}{r}\frac{dF}{dr} + \left(\pm\gamma^2 - \frac{n^2}{r^2}\right)F = 0 \tag{16.81}$$

Through introduction of the nondimensional variable

$$\xi = \gamma r \tag{16.82}$$

Eq. (16.81) is reduced to Bessel's equation

$$\frac{d^2\hat{F}}{d\xi^2} + \frac{1}{\xi}\frac{d\hat{F}}{d\xi} + \left(\pm 1 - \frac{n^2}{\xi^2}\right)\hat{F} = 0 \tag{16.83}$$

where $\hat{F}(\xi) = F(\xi/\gamma)$. A general solution of Eq. (16.83) is

$$\begin{aligned}\hat{F}(\xi) &= AJ_n(\xi) + BY_n(\xi) + CI_n(\xi) + DK_n(\xi)\\ &= AJ_n(\gamma r) + BY_n(\gamma r) + CI_n(\gamma r) + DK_n(\gamma r)\end{aligned} \tag{16.84}$$

where J_n and Y_n are Bessel functions of the first and second kind, and I_n and K_n are modified Bessel functions of the first and second kind, and A, B, C, and D are constants to

be determined. Because Y_n and K_n are singular at $\xi = 0$, for a circular plate without central hole, B and D must be zero to make sense. Thus, the mode shape of the circular plate is expressed by

$$W(r,\theta) = [AJ_n(\gamma r) + CI_n(\gamma r)]\cos n\theta, \quad n = 0,1,2,\ldots \tag{16.85}$$

By the properties of Bessel functions, $J_n(0) = I_n(0) = 0$ for $n > 0$, which implies that $W(0,\theta) = 0$ for $n = 1,2,\ldots$

Equation (16.85), when used with the boundary conditions, leads to a characteristic equation or frequency equation, the roots of which are the natural frequencies of the plate. The characteristic equation for the three types of boundary conditions is presented as follows.

Plate with Clamped Edge Application of the boundary conditions (16.74b) to Eq. (16.85) gives

$$\begin{bmatrix} J_n(\gamma a) & I_n(\gamma a) \\ J_n'(\gamma a) & I_n'(\gamma a) \end{bmatrix} \begin{pmatrix} A \\ C \end{pmatrix} = 0 \tag{16.86}$$

where $J_n' = dJ_n/d\xi$ and $I_n' = dI_n/d\xi$. Vanishing the determinant of the matrix in Eq. (16.86) gives the characteristic equation of the plate

$$J_n(\gamma a)I_n'(\gamma a) - I_n(\gamma a)J_n'(\gamma a) = 0 \tag{16.87}$$

which, via the identities

$$\begin{aligned} \xi J_n'(\xi) &= nJ_n(\xi) - \xi J_{n+1}(\xi) \\ \xi I_n'(\xi) &= nI_n(\xi) + \xi I_{n+1}(\xi) \end{aligned} \tag{16.88}$$

is converted to

$$J_n(\gamma a)I_{n+1}(\gamma a) + I_n(\gamma a)J_{n+1}(\gamma a) = 0 \tag{16.89}$$

The roots γ_{mn} of Eq. (16.89), $m = 0,1,2,\ldots$, are related to the natural frequencies of the plate by

$$\omega_{mn} = \gamma_{mn}^2\sqrt{\frac{D}{\rho}} \tag{16.90}$$

where Eq. (16.79) has been used. The mode shape associated with γ_{mn} is given by

$$W_{mn}(r,\theta) = A\left[J_n(\gamma_{mn}r) - \frac{J_n(\gamma_{mn}a)}{I_n(\gamma_{mn}a)}I_n(\gamma_{mn}r)\right]\cos n\theta \tag{16.91}$$

Letting $W_{mn}(r,\theta)$ be zero defines nodal lines of the mode shape. There are two types of nodal lines: nodal circles determined by $F(r) = 0$, and nodal diameters defined by $\cos n\theta = 0$. As can be shown, the number of nodal circles excluding the plate boundary is m; the number of nodal diameters is n. In the case of zero nodal diameters ($n = 0$), the mode shapes are axisymmetric. The concept of nodal lines discussed here is also applicable to plates with other types of boundary conditions.

Plate with Simply Supported Edge By Eq. (16.85), the boundary conditions (16.74a) become

$$\begin{bmatrix} J_n(\gamma a) & I_n(\gamma a) \\ \phi(\gamma a) & \psi(\gamma a) \end{bmatrix} \begin{pmatrix} A \\ C \end{pmatrix} = 0 \tag{16.92}$$

where

$$\begin{aligned} \phi(\gamma a) &= (\gamma a)^2 J_n''(\gamma a) + \nu(\gamma a) J_n'(\gamma a) \\ \psi(\gamma a) &= (\gamma a)^2 I_n''(\gamma a) + \nu(\gamma a) I_n'(\gamma a) \end{aligned} \tag{16.93}$$

The characteristic equation is

$$J_n(\gamma a)\psi(\gamma a) - I_n(\gamma a)\phi(\gamma a) = 0. \tag{16.94}$$

Through recursive use of the identities (16.88), and the identities

$$\begin{aligned} \xi J_{n+2}(\xi) &= 2(n+1)J_{n+1}(\xi) - \xi J_n(\xi) \\ \xi I_{n+2}(\xi) &= -2(n+1)I_{n+1}(\xi) + \xi J_n(\xi) \end{aligned} \tag{16.95}$$

Eq. (16.94) is reduced to

$$2\gamma a J_n(\gamma a) I_n(\gamma a) - (1-\nu)\left[J_n(\gamma a) I_{n+1}(\gamma a) + J_{n+1}(\gamma a) I_n(\gamma a)\right] = 0 \tag{16.96}$$

The natural frequencies of the plate are given by Eq. (16.90), with γ_{mn} being a root of Eq. (16.96). The expression for the mode shape is the same as given in Eq. (16.91).

Plate with Free Edge By Eq. (16.85), the boundary conditions (16.74c) are written as

$$\begin{bmatrix} \phi(\gamma a) & \psi(\gamma a) \\ \phi_1(\gamma a) & \psi_1(\gamma a) \end{bmatrix} \begin{pmatrix} A \\ C \end{pmatrix} = 0 \tag{16.97}$$

where $\phi(\gamma a)$ and $\psi(\gamma a)$ are given in (16.93), and

$$\begin{aligned} \phi_1(\gamma a) &= (\gamma a)^3 J_n'''(\gamma a) + (\gamma a)^2 J_n''(\gamma a) - \left((2-\nu)n^2 + 1\right)\gamma a J_n'(\gamma a) + (3-\nu)n^2 J_n(\gamma a) \\ \psi_1(\gamma a) &= (\gamma a)^3 I_n'''(\gamma a) + (\gamma a)^2 I_n''(\gamma a) - \left((2-\nu)n^2 + 1\right)\gamma a I_n'(\gamma a) + (3-\nu)n^2 I_n(\gamma a) \end{aligned} \tag{16.98}$$

The characteristic equation is

$$\phi(\gamma a)\psi_1(\gamma a) - \psi(\gamma a)\phi_1(\gamma a) = 0 \tag{16.99}$$

which, by the identities (16.88) and (16.95), is reduced to (Reference 3)

$$\gamma a\left[J_0(\gamma a) I_1(\gamma a) + J_1(\gamma a) I_0(\gamma a)\right] - 2(1-\nu) J_1(\gamma a) I_1(\gamma a) = 0 \tag{16.100a}$$

for $n = 0$, and

$$\begin{aligned} &\left((\gamma a)^4 + n^2(n^2-1)(1-\nu)^2\right)\left\{J_{n-1}(\gamma a) I_{n+1}(\gamma a) + J_{n+1}(\gamma a) I_{n-1}(\gamma a)\right\} \\ &\quad - 2n(1-\nu)(\gamma a)^2\left\{(n-1)J_{n-1}(\gamma a) I_{n-1}(\gamma a) + (n+1)J_{n+1}(\gamma a) I_{n+1}(\gamma a)\right\} = 0 \end{aligned} \tag{16.100b}$$

for $n > 0$. The above equations have an infinite number of roots, γ_{mn}, $m = 0, 1, 2, \ldots$, $n = 0, 1, 2, \ldots$ The eigenfunction (mode shape) corresponding to a root γ_{mn} is given by

$$W_{mn}(r, \theta) = [J_n(\gamma_{mn} r) - \sigma_{mn} I_n(\gamma_{mn} r)] \cos n\theta \tag{16.101}$$

where the coefficient

$$\sigma_{mn} = \frac{n(n-1)(1-\nu)J_n(\gamma_{mn}a) - \gamma_{mn}a(2n+1+\nu)J_{n+1}(\gamma_{mn}a) + (\gamma_{mn}a)^2 J_{n+2}(\gamma_{mn}a)}{n(n-1)(1-\nu)I_n(\gamma_{mn}a) + \gamma_{mn}a(2n+1+\nu)I_{n+1}(\gamma_{mn}a) + (\gamma_{mn}a)^2 I_{n+2}(\gamma_{mn}a)} \tag{16.102}$$

From Eqs. (16.100), the plate with free edge has two zero eigenvalues, $\gamma_{00} = 0$ and $\gamma_{01} = 0$, representing rigid-body modes.

Solution by the Toolbox The Toolbox of this chapter provides a function `cirplate` for computing eigensolutions of circular plates; see Window 4.1.

Window 4.1. Function `cirplate`

MATLAB Function: **`cirplate`**

Purpose:
To compute the eigensolutions of a circular plate.

Synopsis:

```
cirplate(Plate_Pars, BC_Spec)
cirplate(Plate_Pars, BC_Spec, mn_mode)
```

Description:
`cirplate(Plate_Pars, BC_Spec)` computes the natural frequencies and mode shapes of the plate, where `Plate_Pars` is a vector of plate parameters

```
Plate_Pars = [rou E mu a h]
```

with `rou` = mass density (mass per unit area), `E` = Young's modulus, `mu` = Poisson's ratio, `a` = radius, and `h` = thickness; `BC_Spec` is an integer specifying the boundary condition with the options

```
BC_Spec = 0  for a free edge
BC_Spec = 1  for a simply-supported edge
BC_Spec = 2  for a clamped edge
```

`cirplate(Plate_Pars, BC_Spec, mn_mode)` computes the eigensolutions of the plate with `mn_mode = [m n]`, where `m` is the highest number of nodal circles and `n` is the highest number of nodal diameters. By default, `mn_mode = [4 4]`.

Note. Function `cirplate` displays computed eigenvalues in the following forms:

ω_{mn}, natural frequency in rad/s

γ_{mn}, frequency parameter

$\xi_{mn} = \gamma_{mn}a$, nondimensional frequency parameter

According to Eqs. (16.89), (16.96), and (16.100), ξ_{mn} is independent of Poisson's ratio ν for plates with clamped edges, and is dependent upon ν for plates with simply-supported or free edges.

After `cirplate` is executed, function `systinfo` can be called at any time, to get access to the information on plate parameters, boundary conditions, and natural frequencies. Also, function `plotmsh4` (Window 2.5) can be used to plot the mode shapes of the plate and to display the mathematical expressions of the mode shapes in terms of Bessel functions. The Bessel functions J_n and I_n in Eqs. (16.91) and (16.101) can be evaluated by standard MATLAB functions `besselj(n,z)` and `besseli(n,z)`. Additionally, function `animode4` (Window 2.6) can be used to animate the vibration of the plate in a particular mode.

EXAMPLE 4.1

A circular plate with free edge has parameters

$$\rho = 0.01, \quad E = 1.5 \times 10^4, \quad \nu = 0.3, \quad a = 0.8, \quad h = 0.01$$

```
>> cirplate([0.01 1.5e4 0.3 0.8 0.01], 0)
```

gives the natural frequencies of the plate as follows:

```
Natural Frequencies:
--------------------
 w_mn      n = 0      1         2         3         4
-------------------------------------------------------
m =
     0         0         0    3.3151    7.5739   13.1635
1.0000    5.2137   11.8568   20.4199   30.7044   42.6085
2.0000   22.2625   34.6370   48.8565   64.8283   82.4851
3.0000   50.8162   68.8883   88.7796  110.4299  133.7908
4.0000   90.8136  114.5878  140.1633  167.4979  196.5545

Here  w_mn = natural frequency rad/s
        m = number of nodal circles
        n = number of nodal diameters
```

Through use of `plotmsh4(m,n)`, the locations of nodal circles of the mode shapes are collected in Table 16.4.1. (An example of using `plotmsh4` to plot mode shapes is given in Section 16.1, Getting Started.)

Also, to animate mode (0, 3), which has zero nodal circle and 3 nodal diameters, type the following commands

```
>> T = 2*pi/7.5739;
>> animode4(0, 3, 3, [ ], T, 7, 1)
```

where T is the period of the mode, namely, $T = 2\pi/\omega_{03} = 0.8296$. The first six frames of the animation are given in Fig. 16.4.3.

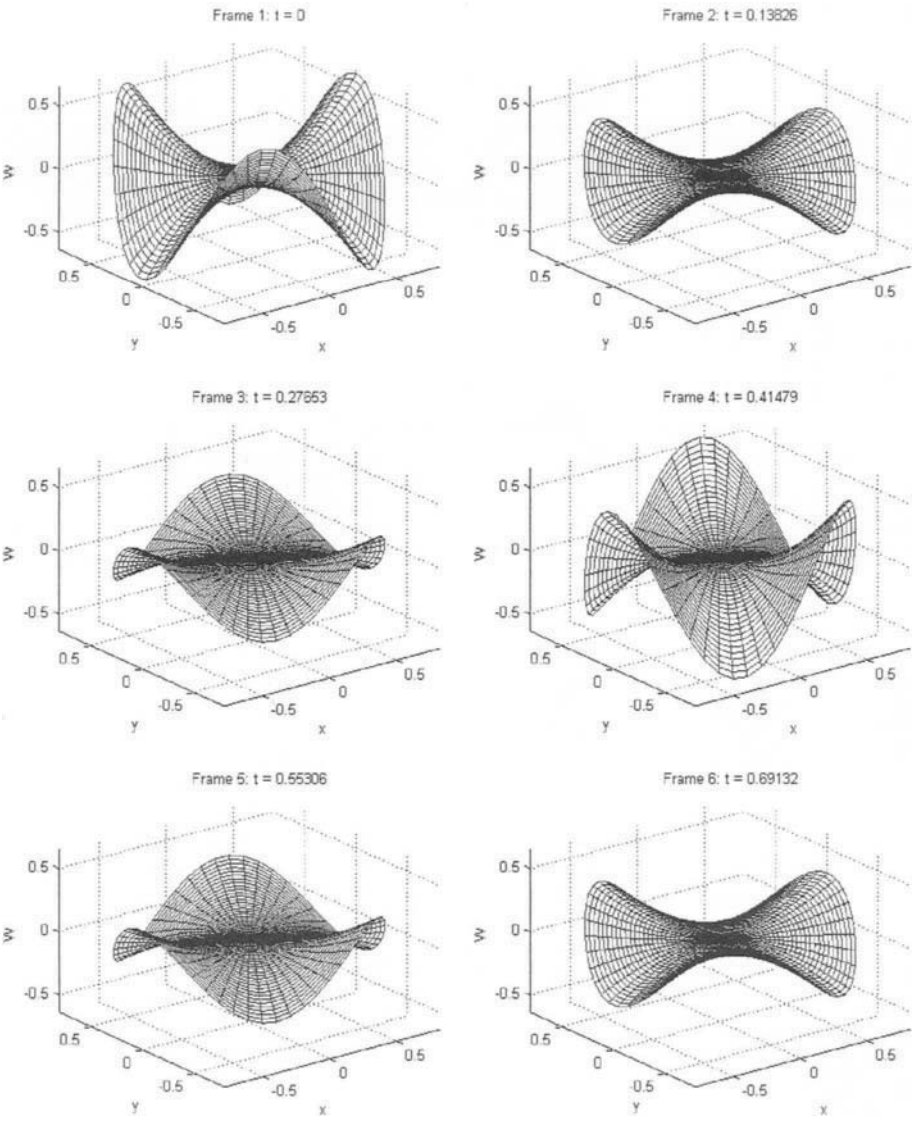


FIGURE 16.4.3 Frames of the animated plate vibration mode (0,3).

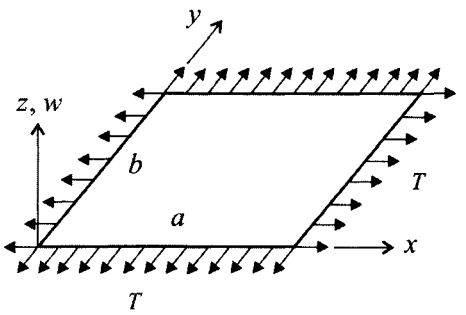
TABLE 16.4.1 Location of Nodal Circles in r

	$n = 0$	1	2	3	4
$m = 1$	0.5440	0.6260	0.6580	0.6780	0.6900
2	0.3120	0.3973	0.4480	0.4827	0.5087
	0.6733	0.6973	0.7107	0.7200	0.7267
3	0.2054	0.2811	0.3318	0.3696	0.3994
	0.4720	0.5150	0.5440	0.5660	0.5824
	0.7150	0.7260	0.7340	0.7400	0.7440
4	0.1536	0.2179	0.2641	0.3001	0.3295
	0.3526	0.3990	0.4328	0.4592	0.4805
	0.5536	0.5792	0.5976	0.6128	0.6241
	0.7360	0.7424	0.7472	0.7512	0.7544

16.5 Quick Solution Guide

System Description

Rectangular membranes



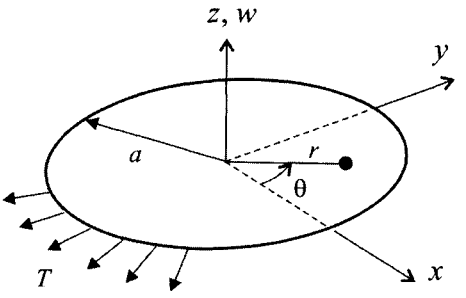
$$w(x,y,t) = W(x,y)\cos(\omega t + \alpha)$$

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) W(x,y) + \frac{\rho\omega^2}{T} W(x,y) = 0$$

ω – natural frequency

$W(x,y)$ – mode shape

Circular membranes



$$w(r,\theta,t) = W(r,\theta)\cos(\omega t + \alpha)$$

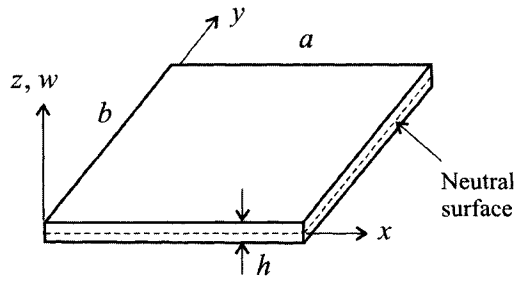
$$\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2}\right) W(r,\theta) + \frac{\rho\omega^2}{T} W(r,\theta) = 0$$

ω – natural frequency

$W(r,\theta)$ – mode shape

(continued)

Rectangular plates



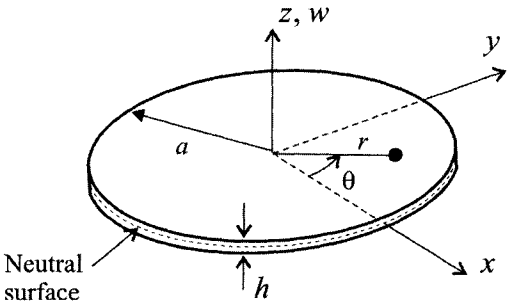
$$w(x,y,t) = W(x,y)\cos(\omega t + \alpha)$$

$$\left(\frac{\partial^4}{\partial x^4} + 2\frac{\partial^2}{\partial x^2}\frac{\partial^2}{\partial y^2} + \frac{\partial^4}{\partial y^4}\right) W(x,y) = \frac{\rho\omega^2}{D} W(x,y)$$

ω – natural frequency

$W(x,y)$ – mode shape

Circular plates



$$w(r,\theta,t) = W(r,\theta)\cos(\omega t + \alpha)$$

$$\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2}\right)^2 W(r,\theta) = \frac{\rho\omega^2}{D} W(r,\theta)$$

ω – natural frequency

$W(r,\theta)$ – mode shape

Problem to Solve

To compute the natural frequencies and mode shapes of tensioned membranes and thin plates, and to animate the vibration of those continua in particular modes.

MATLAB Functions

This chapter has a set (toolbox) of MATLAB functions for free vibration analysis of membranes and plates; see the table below. Refer to the corresponding window or section for the utility of a particular function.

Free Vibration of Membranes		
`recmemb`	Compute eigensolutions of a rectangular membrane	Window 2.1
`plotmsh1`	Plot mode shapes of a rectangular membrane	Window 2.2
`animode1`	Animate modes of vibration of a rectangular membrane	Window 2.3
`cirmemb`	Compute eigensolutions of a circular membrane	Window 2.4
`plotmsh2`	Plot mode shapes of a circular membrane	Window 2.5
`animode2`	Animate modes of vibration of a circular membrane	Window 2.6
Free Vibration of Rectangular Plates—Navier and Lévy Solutions		
`recplateLV`	Compute Navier- or Lévy-type eigensolutions of a rectangular plate with at least two opposite edges simply supported	Window 3.1
`plotmsh3`	Plot mode shapes of a rectangular plate	Window 2.2
`animode3`	Animate modes of vibration of a rectangular plate	Window 2.3

Free Vibration of Rectangular Plates—Finite Element Solutions		
recplateFE	Compute eigensolutions of a rectangular plate with general boundary conditions	Window 3.2
plotmsh3f	Plot mode shapes of a rectangular plate	Window 3.3
animode3f	Animate modes of vibration of a rectangular plate	Window 3.4
Free Vibration of Circular Plates		
cirplate	Compute eigensolutions of a circular plate	Window 4.1
plotmsh4	Plot mode shapes of a circular plate	Window 2.5
animode4	Animate modes of vibration of a circular plate	Window 2.6
Utilities		
systinfo	Display the information and eigensolutions of a membrane or plate	Sections 16.2.1, 16.2.2, 16.3.3, 16.4.2
TBdemo	Show how the Toolbox works and what it can do	Section 16.1
TBinfo	Show the information of the functions in the Toolbox	Section 16.1
RunEx	Run all the numerical examples contained in this chapter	Section 16.1

Solution Procedure

Rectangular membranes

To compute eigensolutions, type

```
>> recmemb(Memb_Pars, BC_Spec)
```

where `Memb_Pars = [rou T a b]`, and `BC_Spec = [J1 J2 J3 J4]` with `Jk = 0` or 1.

To plot mode shapes, type

```
>> plotmsh1(m,n)
```

To animate modes of vibration, type

```
>> animode1(m,n)
```

Circular membranes

To compute eigensolutions, type

```
>> cirmemb(Memb_Pars, BC_Spec)
```

where `Memb_Pars = [rou T a]` and `BC_Spec = 0` or 1.

To plot and animate modes of vibration, use `plotmsh2(m,n)` and `animode2(m,n)`.

Rectangular plates

To compute Navier- or Lévy-type eigensolutions, type

```
>> recplateLV(Plate_Pars, BC_Spec)
```

where `Plate_Pars = [rou E mu a b h]`, and `BC_Spec = [B0 Bb]` with `B0, Bb = 0` or 1 or 2. To plot and animate a mode, use `plotmsh3(m,n)` and `animode3(m,n)`.

To compute eigensolutions via the finite element method, type

```
>> recplateFE(Plate_Pars, BC_Spec)
```

where `Plate_Pars = [rou E mu a b h]` and `[J1 J2 J3 J4]` with `Jk = 0` or 1 or 2. The computed modes can be plotted and animated by functions `plotmsh3f(n)` and `animode3f(n)`.

Circular plates
To compute eigensolutions, type
```
>> cirplate(Plate_Pars, BC_Spec)
```
where `Plate_Pars = [rou E mu a h]` and `BC_Spec = 0` or 1 or 2.
To plot and animate modes of vibration, use `plotmsh4(m,n)` and `animode4(m,n)`.

16.6 References

References on Plate Theory

1. Gould P L 1998 *Analysis of Shells and Plates*, 2nd edition, Prentice-Hall: Upper Saddle River, New Jersey.
2. Leissa A W 1969 *Vibration of Plates*, NASA SP-160, U.S. Government Printing Office: Washington, D.C.
3. Rao J S 1999 *Dynamics of Plates*, Narosa Publishing House: New Delhi, India.
4. Soedel W 1993 *Vibrations of Shells and Plates*, 2nd edition, Marcel Dekker, Inc.: New York.
5. Timoshenko S 1940 *Theory of Plates and Shells*, McGraw-Hill: New York.

References on Finite Element Methods

6. Cook R D, Malkus D S, Plesha M E, Witt R J 2001 *Concepts and Applications of Finite Element Analysis*, 4th edition, John Wiley & Sons, Inc.: New York.
7. Hughes T J R 2000 *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Publications: New York.
8. Petyt M 1990 *Introduction to Finite Element Vibration Analysis*, Cambridge University Press: Cambridge.
9. Reddy J N 1993 *Introduction to the Finite Element Method*, 2nd edition, McGraw-Hill Book Company: New York.
10. Zienkiewicz O C, Taylor R L 2000 *The Finite Element Method*, 5th edition, Butterworth-Heinemann: Oxford.

This Page Intentionally Left Blank

Appendix A Commonly Used Mathematical Formulas

Inside
- Algebraic Formulas
- Areas and Volumes of Common Shapes
- Trigonometry
- Hyperbolic Functions
- Derivatives and Integration
- Series Expansion
- Analytical Geometry
- Vector Analysis
- Matrix Theory
- Complex Numbers and Complex Functions
- Laplace Transforms
- Inverse Laplace Transform via Partial Fraction Expansion

A.1 Algebraic Formulas

Factors and Expansions

$$(a+b)^2 = a^2 + 2ab + b^2$$

$$(a-b)^2 = a^2 - 2ab + b^2$$

$$a^2 - b^2 = (a+b)(a-b)$$

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

$$(a-b)^3 = a^3 - 3a^2b + 3ab^2 - b^3$$

$$a^3 + b^3 = (a + b)(a^2 - ab + b^2)$$

$$a^3 - b^3 = (a - b)(a^2 + ab + b^2)$$

$$(a + b + c)^2 = a^2 + b^2 + c^2 + 2ab + 2ac + 2bc$$

$$(a + b)^n = C_n^0 a^n + C_n^1 a^{n-1} b + C_n^2 a^{n-2} b^2 + \cdots + C_n^{n-1} a b^{n-1} + C_n^n b^n$$

where $C_n^j = \dfrac{n!}{(n-j)!\,j!}$, with $C_n^0 = 1$.

Quadratic Equations

The quadratic equation

$$ax^2 + bx + c = 0$$

has the roots

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

If $b^2 - 4ac \geq 0$, the roots are real.

If $b^2 - 4ac < 0$, the roots are complex.

If $b = 0$ and $ac > 0$, the roots are imaginary.

Logarithms

Definition Given a positive constant a, for a positive number x, its logarithm to the base a is expressed by $y = \log_a x$, which implies $x = a^y$.

Properties of logarithms:

$$\log_a(xy) = \log_a(x) + \log_a(y)$$

$$\log_a\left(\frac{x}{y}\right) = \log_a(x) - \log_a(y)$$

$$\log_a\left(x^n\right) = n \log_a(x)$$

Two commonly used bases for logarithms:

(i) Common logarithms—logarithms to the base 10, $\log_{10} x$.
(ii) Natural logarithms—logarithms to the base e, $\log_e x$, where e is an irrational number given by $e = 2.718281828459\cdots$. The $\log_e x$ is usually written as $\ln x$.

The conversion between common and natural logarithms:

$$\log_{10} x = \frac{1}{\ln 10} \ln x, \quad \ln x = \frac{1}{\log_{10} e} \log_{10} x$$

Sums of Powers of Integers

$$1 + 2 + 3 + \cdots + n = \frac{1}{2}n(n+1)$$

$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{1}{6}n(n+1)(2n+1)$$

$$1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{1}{4}n^2(n+1)^2$$

$$1 + 3 + 5 + \cdots + (2n-1) = n^2$$

$$2 + 4 + 6 + \cdots + 2n = n(n+1)$$

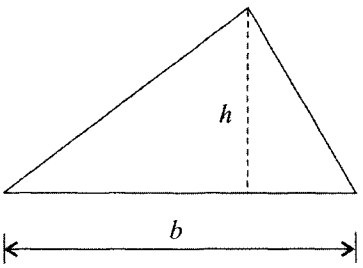
$$1^2 + 3^2 + 5^2 + \cdots + (2n-1)^2 = \frac{1}{3}n(4n^2 - 1)$$

$$1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + \cdots + n(n+1) = \frac{1}{3}n(n+1)(n+2)$$

A.2 Areas and Volumes of Common Shapes

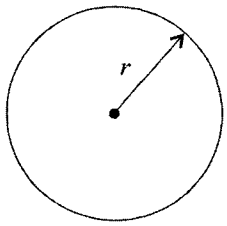
TABLE A1 Areas of Planar Shapes

Triangle



Area $A = \frac{1}{2}bh$

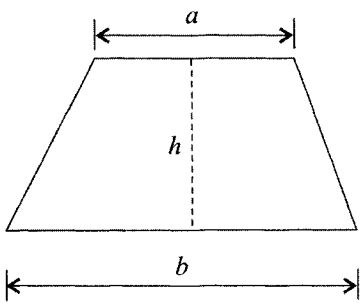
Circle



Area $A = \pi r^2$

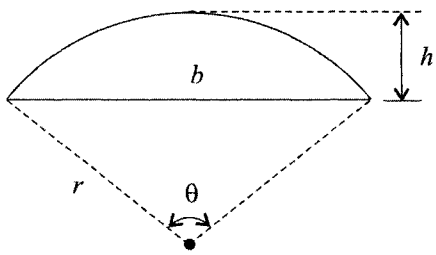
Circumference $C = 2\pi r$

Trapezoid



Area $A = \frac{1}{2}(a+b)h$

Segment of a circle



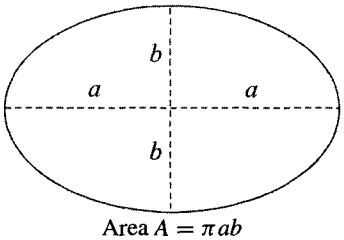
Area $A = \frac{1}{2}r^2(\theta - \sin\theta)$, θ in rad

$b = 2r\sin(\theta/2)$, $h = 2r\sin^2(\theta/4)$

(continued)

TABLE A1 Areas of Planar Shapes *(continued)*

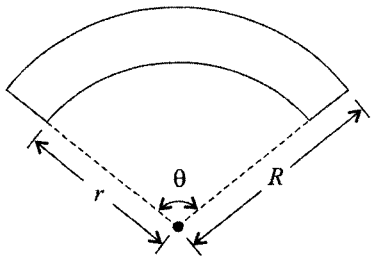
Ellipse



Area $A = \pi ab$

Circumference $C \approx 2\pi\sqrt{\dfrac{a^2 + b^2}{2}}$

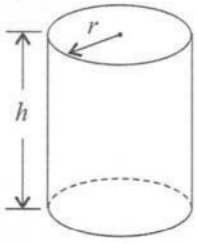
Annular sector



Area $A = \dfrac{1}{2}\theta\,(R^2 - r^2)$, θ in rad

TABLE A2 Volumes of Three-Dimensional Shapes

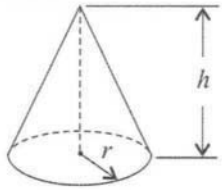
Circular cylinder



Volume $V = \pi r^2 h$

Total surface $S = 2\pi r(h + r)$

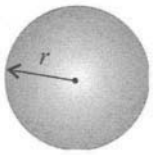
Circular cone



Volume $V = \dfrac{1}{3}\pi r^2 h$

Total surface $S = \pi r\left(r + \sqrt{r^2 + h^2}\right)$

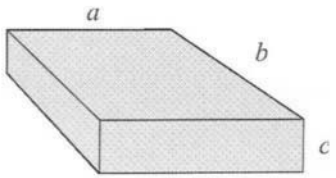
Sphere



Volume $V = \dfrac{4}{3}\pi r^3$

Surface $S = 4\pi r^2$

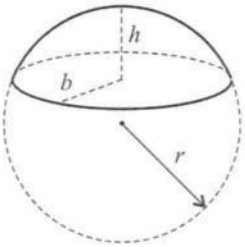
Rectangular prism



Volume $V = abc$

Total surface $S = 2(ab + bc + ca)$

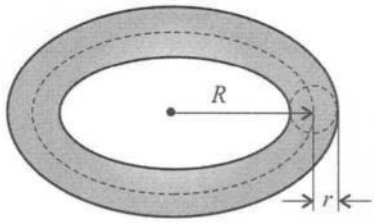
Segment of a sphere



Volume $V = \dfrac{1}{6}\pi h(3b^2 + h^2) = \dfrac{1}{3}\pi h^2(3r - h)$

Total surface $S = \pi(2rh + b^2)$

Hoop (ring)



Volume $V = 2\pi^2 Rr^2$

Surface $S = 4\pi^2 Rr$

A.3 Trigonometry

Units of Angles

Angles are usually measured in degrees (°) and radians (rad). For an angle measured by θ in degrees and by α in radians,

$$\alpha = \frac{\theta}{180} \times \pi \text{ rad} \quad \text{or} \quad \theta = \frac{\alpha}{\pi} \times 180^\circ$$

where π is an irrational number given by $\pi = 3.14159265\ldots$. Thus,

$$1 \text{ rad} = 57.29577951^\circ$$
$$1^\circ = 0.0174532925 \text{ rad}$$
$$\pi \text{ rad} = 180^\circ$$

Trigonometric Functions

In Fig. A1, angle θ is made by line OP and the x-axis of the Cartesian coordinate system Oxy. The trigonometric functions of θ are defined as follows

$$\sin\theta = \frac{y}{r}, \quad \cos\theta = \frac{x}{r}, \quad \tan\theta = \frac{y}{x}$$
$$\csc\theta = \frac{r}{y}, \quad \sec\theta = \frac{r}{x}, \quad \cot\theta = \frac{x}{y}$$

where x and y can be any real numbers, and $r = |OP| = \sqrt{x^2 + y^2}$. The values of trigonometric functions at some specific angles are shown below.

θ	$\sin\theta$	$\cos\theta$	$\tan\theta$	$\cot\theta$
0°	0	1	0	∞
30°	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$	$\sqrt{3}$
45°	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1	1
60°	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$	$\frac{\sqrt{3}}{3}$
90°	1	0	∞	0

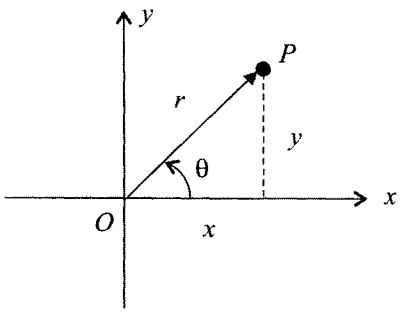


FIGURE A1

Commonly Used Formulas

Identities

$$\cot\theta = \frac{1}{\tan\theta}, \quad \csc\theta = \frac{1}{\sin\theta}, \quad \sec\theta = \frac{1}{\cos\theta}, \quad \tan\theta = \frac{\sin\theta}{\cos\theta}, \quad \cot\theta = \frac{\cos\theta}{\sin\theta}$$

$$\sin^2\theta + \cos^2\theta = 1, \quad 1 + \tan^2\theta = \sec^2\theta, \quad 1 + \cot^2\theta = \csc^2\theta$$

$$e^{i\theta} = \cos\theta + i\sin\theta, \quad \sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i}, \quad \cos\theta = \frac{e^{i\theta} + e^{-i\theta}}{2}, \quad \text{with} \quad i = \sqrt{-1}$$

Functions of angle sum and difference

$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$$

$$\cos(\alpha \pm \beta) = \sin\alpha\sin\beta \mp \cos\alpha\cos\beta$$

$$\tan(\alpha \pm \beta) = \frac{\tan\alpha \pm \tan\beta}{1 \mp \tan\alpha\tan\beta}$$

Example:

$$A\sin\theta + B\cos\theta = M\sin(\theta + \alpha) = M\sin(\theta - \beta)$$

where $M = \sqrt{A^2 + B^2}$, $\alpha = \tan^{-1}(A/B)$, $\beta = \tan^{-1}(B/A)$

Functions of double angles

$$\sin 2\theta = 2\sin\theta\cos\theta$$

$$\cos 2\theta = \cos^2\theta - \sin^2\theta = 1 - 2\sin^2\theta = 2\cos^2\theta - 1$$

$$\tan 2\theta = \frac{2\tan\theta}{1 - \tan^2\theta}$$

Products of functions

$$\sin\alpha\sin\beta = -\frac{1}{2}(\cos(\alpha + \beta) - \cos(\alpha - \beta))$$

$$\cos\alpha\cos\beta = \frac{1}{2}(\cos(\alpha + \beta) + \cos(\alpha - \beta))$$

$$\sin\alpha\cos\beta = \frac{1}{2}(\sin(\alpha + \beta) + \sin(\alpha - \beta))$$

$$\cos\alpha\sin\beta = \frac{1}{2}(\sin(\alpha + \beta) - \sin(\alpha - \beta))$$

Sums and differences of functions

$$\sin\alpha + \sin\beta = 2\sin\frac{\alpha + \beta}{2}\cos\frac{\alpha - \beta}{2}$$

$$\sin\alpha - \sin\beta = 2\cos\frac{\alpha + \beta}{2}\sin\frac{\alpha - \beta}{2}$$

$$\cos\alpha + \cos\beta = 2\cos\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2}$$

$$\cos\alpha - \cos\beta = -2\sin\frac{\alpha+\beta}{2}\sin\frac{\alpha-\beta}{2}$$

Relations for Right Triangles

For the right triangle in Fig. A2

$$\sin A = \frac{a}{c}, \quad \cos A = \frac{b}{c}, \quad \tan A = \frac{a}{b}$$

$$a^2 + b^2 = c^2$$

Relations for Oblique Triangles

For the oblique triangle in Fig. A3, the following relations can be used for solution.

Law of sines

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$$

Law of cosines

$$a^2 = b^2 + c^2 - 2bc\cos A, \quad b^2 = c^2 + a^2 - 2ca\cos B, \quad c^2 = a^2 + b^2 - 2ab\cos C$$

Example. Given sides a and b and included angle C, side c and angles A and B are determined by

$$c = \sqrt{a^2 + b^2 - 2ab\cos C}, \quad A = \sin^{-1}\left(\frac{a}{c}\sin C\right), \quad B = 180^\circ - (C + A).$$

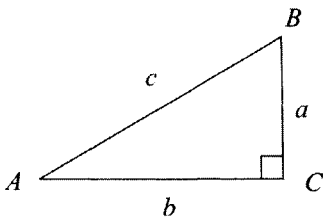


FIGURE A2

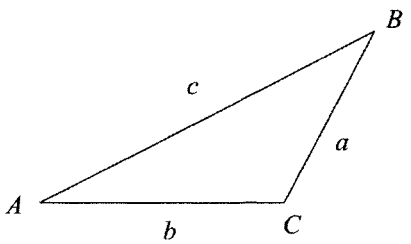


FIGURE A3

A.4 Hyperbolic Functions

Definition

Hyperbolic sine: $$\sinh x = \frac{e^x - e^{-x}}{2}$$

Hyperbolic cosine: $$\sinh x = \frac{e^x + e^{-x}}{2}$$

Hyperbolic tangent: $$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Commonly Used Formulas

Identities

$$\tanh x = \frac{\sinh x}{\cosh x}$$

$$\cosh^2 x - \sinh^2 x = 1$$

Angle-sum/difference relations

$$\sinh(\alpha \pm \beta) = \sinh\alpha \cosh\beta \pm \cosh\alpha \sinh\beta$$

$$\cosh(\alpha \pm \beta) = \sinh\alpha \sinh\beta \pm \cosh\alpha \cosh\beta$$

$$\tanh(\alpha \pm \beta) = \frac{\tanh\alpha \pm \tanh\beta}{1 \pm \tanh\alpha \tanh\beta}$$

Function-sum/difference relations

$$\sinh\alpha + \sinh\beta = 2\sinh\frac{\alpha+\beta}{2}\cosh\frac{\alpha-\beta}{2}$$

$$\sinh\alpha - \sinh\beta = 2\cosh\frac{\alpha+\beta}{2}\sinh\frac{\alpha-\beta}{2}$$

$$\cosh\alpha + \cosh\beta = 2\cosh\frac{\alpha+\beta}{2}\cosh\frac{\alpha-\beta}{2}$$

$$\cosh\alpha - \cosh\beta = 2\sinh\frac{\alpha+\beta}{2}\sinh\frac{\alpha-\beta}{2}$$

A.5 Derivatives and Integration

Derivatives

Assume u and v are functions of x.

$$\frac{d}{dx}(uv) = \frac{du}{dx}v + u\frac{dv}{dx}$$

$$\frac{d}{dx}\left(\frac{u}{v}\right) = \frac{1}{v^2}\left(v\frac{du}{dx} - u\frac{dv}{dx}\right)$$

$$\frac{d}{dx}[f(u)] = \frac{df}{du}\cdot\frac{du}{dx}$$

TABLE A3 Derivatives of Basic Functions

$f(x)$	$\frac{d}{dx}f(x)$	$f(x)$	$\frac{d}{dx}f(x)$
x^n	nx^{n-1}	$\sin^{-1}(x)$	$\frac{1}{\sqrt{1-x^2}}$
$\ln x$	$\frac{1}{x}$	$\cos^{-1}(x)$	$-\frac{1}{\sqrt{1-x^2}}$
e^x	e^x	$\tan^{-1}(x)$	$\frac{1}{1+x^2}$
a^x	$a^x \ln a$	$\cot^{-1}(x)$	$-\frac{1}{1+x^2}$
$\log_a x$	$\frac{1}{x}\log_a e$	$\sinh x$	$\cosh x$
$\sin x$	$\cos x$	$\cosh x$	$\sinh x$
$\cos x$	$-\sin x$	$\tanh x$	$\frac{1}{\cosh^2 x}$
$\tan x$	$\sec^2 x$	x^x	$x^x(1+\ln x)$
$\cot x$	$-\csc^2 x$	$\sqrt{x}$	$\frac{1}{2}\frac{1}{\sqrt{x}}$

Integration

Assume u and v are functions of x.

$$\int udu = uv - \int vdu \quad \text{or} \quad \int u\frac{dv}{dx}dx = uv - \int \frac{du}{dx}vdx \quad \text{(integral by part)}$$

$$\int \frac{d}{du}[f(u(x))]\,d[u(x)] = f(u(x))$$

$$\int adx = ax \quad (a \text{ constant})$$

TABLE A4 Integrals of Basic Functions

$f(x)$	$\int f(x)\,dx$	$f(x)$	$\int f(x)\,dx$
$x^n \quad (n \neq -1)$	$\frac{1}{n+1}x^{n+1}$	$\frac{1}{a^2+x^2} \quad (a>0)$	$\frac{1}{a}\tan^{-1}\left(\frac{x}{a}\right)$
$\frac{1}{x}$	$\ln x$	$\frac{1}{a^2-x^2} \quad (\lvert x\rvert < \lvert a\rvert)$	$\frac{1}{2a}\ln\left(\frac{a+x}{a-x}\right)$
e^{ax}	$\frac{1}{a}e^{ax}$	$\frac{1}{x^2-a^2} \quad (\lvert x\rvert > \lvert a\rvert)$	$\frac{1}{2a}\ln\left(\frac{x+a}{x-a}\right)$
$a^{bx}\ (a>0)$	$\frac{1}{b\ln a}a^{bx}$	$\sinh x$	$\cosh x$
$\ln ax$	$x(\ln ax - 1)$	$\cosh x$	$\sinh x$

(continued)

TABLE A4 Integrals of Basic Functions *(continued)*

$f(x)$	$\int f(x)\,dx$	$f(x)$	$\int f(x)\,dx$
$\sin ax$	$-\frac{1}{a}\cos ax$	$\tanh x$	$\ln(\cosh x)$
$\cos ax$	$\frac{1}{a}\sin ax$	$\frac{1}{\sqrt{a^2-x^2}}$	$\sin^{-1}\left(\frac{x}{\lvert a\rvert}\right)$
$\tan ax$	$-\frac{1}{a}\ln(\cos ax)$	$\frac{1}{\sqrt{a^2\pm x^2}}$	$\ln\left(x+\sqrt{x^2\pm a^2}\right)$
$\sin^2 x$	$\frac{x}{2}-\frac{1}{2}\sin x\cos x$	$e^{ax}\sin bx$	$\frac{e^{ax}}{a^2+b^2}(a\sin bx-b\cos bx)$
$\cos^2 x$	$\frac{x}{2}+\frac{1}{2}\sin x\cos x$	$e^{ax}\cos bx$	$\frac{e^{ax}}{a^2+b^2}(a\cos bx+b\sin bx)$

A.6 Series Expansion

Taylor's Series

$$f(x)=f(a)+f'(a)(x-a)+\frac{f''(a)}{2!}(x-a)^2+\cdots+\frac{f^{(n)}(a)}{n!}(x-a)^n+\cdots$$

where $f'=\dfrac{df}{dx}$, and $f^{(n)}=\dfrac{d^nf}{dx^n}$. Taylor's series can also be given in the increment form

$$f(x+h)=f(x)+hf'(x)+\frac{h^2}{2!}f''(x)+\cdots+\frac{h^n}{n!}f^{(n)}(x)+\cdots$$

Maclaurin's Series ($a=0$ in Taylor's series)

$$f(x)=f(0)+f'(0)x+\frac{f''(0)}{2!}x^2+\cdots+\frac{f^{(n)}(0)}{n!}x^n+\cdots$$

Series of Certain Functions

$$e^x=1+\frac{x}{1!}+\frac{1}{2!}x^2+\frac{1}{3!}x^3+\cdots+\frac{1}{n!}x^n+\cdots \qquad (|x|<\infty)$$

$$\sin x=x-\frac{1}{3!}x^3+\frac{1}{5!}x^5-\frac{1}{7!}x^7+\cdots \qquad (|x|<\infty)$$

$$\cos x=1-\frac{1}{2!}x^2+\frac{1}{4!}x^4-\frac{1}{6!}x^6+\cdots \qquad (|x|<\infty)$$

$$\sinh x=x+\frac{1}{3!}x^3+\frac{1}{5!}x^5+\cdots+\frac{1}{(2n+1)!}x^{2n+1}+\cdots \qquad (|x|<\infty)$$

$$\cosh x = 1 + \frac{1}{2!}x^2 + \frac{1}{4!}x^5 + \cdots + \frac{1}{(2n)!}x^{2n} + \cdots \qquad (|x| < \infty)$$

$$\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \cdots + \frac{(-1)^{n+1}}{n}x^n + \cdots \qquad (-1 < |x| \leq 1)$$

$$\sqrt{1 \pm x} = 1 \pm \frac{1}{2}x - \frac{1}{2}\cdot\frac{1}{4}x^2 \pm \frac{1}{2}\cdot\frac{1}{4}\cdot\frac{3}{6}x^3 - \frac{1}{2}\cdot\frac{1}{4}\cdot\frac{3}{6}\cdot\frac{5}{8}x^4 \pm \cdots \qquad (|x| \leq 1)$$

$$\frac{1}{\sqrt{1 \pm x}} = 1 \mp \frac{1}{2}x + \frac{1}{2}\cdot\frac{3}{4}x^2 \mp \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}x^3 + \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\cdot\frac{7}{8}x^4 \mp \cdots \qquad (|x| < 1)$$

Fourier Series

Consider a periodic function $f(x)$ of period T, i.e., $f(x+T) = f(x)$ for any x. If $f(x)$ is piecewise continuous, with a finite number of jump discontinuities in $[0, T]$, and has a finite number of maxima and minima in $[0, T]$, then it can be represented by the Fourier series

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}\left(a_k \cos\frac{2k\pi x}{T} + b_k \sin\frac{2k\pi x}{T}\right)$$

with the coefficients

$$a_k = \frac{2}{T}\int_0^T f(x)\cos\frac{2k\pi x}{T}dx, \quad k = 0, 1, 2, \ldots$$

$$b_k = \frac{2}{T}\int_0^T F(t)\sin\frac{2k\pi x}{T}dx, \quad k = 1, 2, 3, \ldots$$

The series converges to $f(x)$ at a point where the function is continuous, and to $\frac{1}{2}\{f(x+) + f(x-)\}$ at a point where the function has a jump discontinuity.

If a period function $f(x)$ of period $2L$ satisfies the above-mentioned conditions over the range $[-L, L]$, it can be expanded in the series

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}\left(a_k \cos\frac{k\pi x}{L} + b_k \sin\frac{k\pi x}{L}\right)$$

with

$$a_k = \frac{1}{L}\int_{-L}^{L} f(x)\cos\frac{k\pi x}{T}dx, \quad k = 0, 1, 2, \ldots$$

$$b_k = \frac{1}{L}\int_{-L}^{L} f(x)\sin\frac{k\pi x}{T}dx, \quad k = 1, 2, 3, \ldots$$

For example, for the periodic function $f(x) = |x|$, $-L \leq x \leq L$, its Fourier expansion is

$$f(x) = \frac{L}{2} - \frac{4L}{\pi^2}\sum_{k=1}^{\infty}\frac{1}{(2k-1)^2}\cos\frac{(2k-1)\pi x}{L}$$

The Toolbox of Chapter 10 (Vibration Analysis of One-Degree-of-Freedom Systems) has MATLAB functions for obtaining Fourier series of periodic functions.

A.7 Analytical Geometry

Planar Coordinate Systems

A point P in a plane can be expressed by rectangular coordinates (x, y), or by polar coordinates (r, θ); see Fig. A4. These coordinates are related by

$$x = r\cos\theta, \quad y = r\sin\theta$$

or

$$r = \sqrt{x^2 + y^2}, \quad \theta = \tan^{-1}\left(\frac{y}{x}\right)$$

Straight Lines

$$y = kx + b$$

where k = slope and b = intercept; see Fig. A5.
General form of straight lines:

$$Ax + By + C = 0$$

Circles

A circle of radius r is centered at (a, b); see Fig. A6. The circle is described by

$$(x - a)^2 + (y - b)^2 = r^2$$

General form of circles:

$$A\left(x^2 + y^2\right) + Dx + Ey + F = 0$$

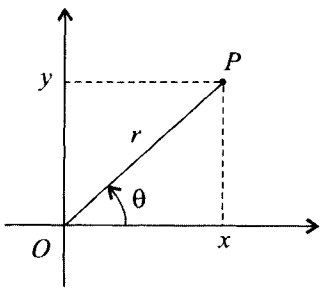


FIGURE A4

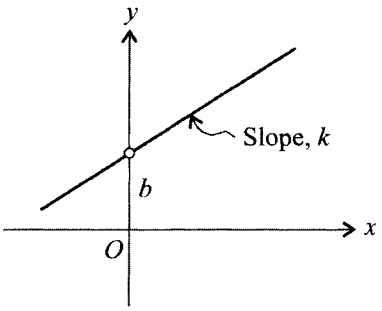


FIGURE A5

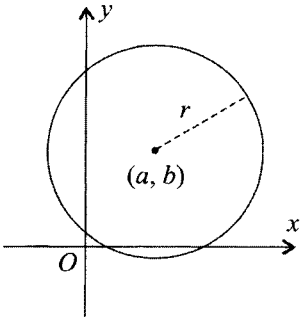


FIGURE A6

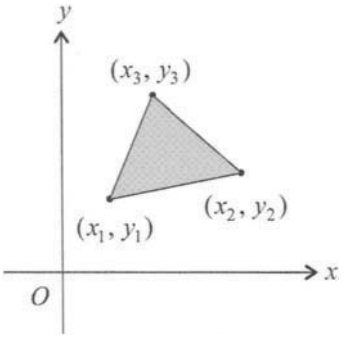


FIGURE A7

Area of a Triangle

For the triangle of corners (x_1, y_1), (x_2, y_2), and (x_3, y_3) in Fig. A7, its area is

$$\text{Area} = \frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}.$$

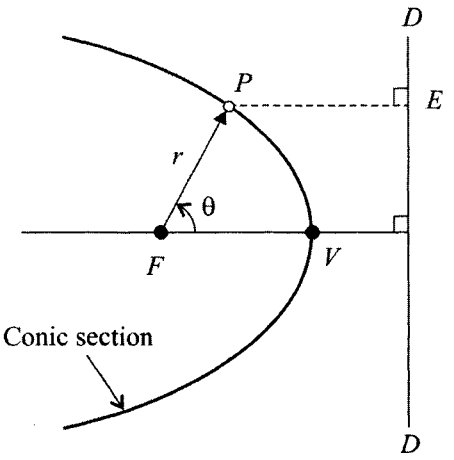


FIGURE A8

Conic Sections

A conic section shown in Fig. A8 is a trajectory of a point P moving in a plane of a fixed point F (called the *focus*) and a fixed line D-D (called the *directrix*) such that the distance ratio

$$e = \frac{|PF|}{|PE|} = \frac{r}{p - r\cos\theta}$$

is a constant (called the *eccentricity*), where $|PF|$ is the distance of P from F, $|PE|$ is the distance of P from line D-D, and p is the distance from the directrix to the focus. Also, point V is a vertex and line V-F is the axis of the conic section. In the polar coordinates centered at F, the equation for a conic section is

$$r = \frac{ep}{1 + e\cos\theta}.$$

Depending on the value of eccentricity e, there are three types of conic sections.

Ellipses ($e < 1$, Fig. A9)

standard equation: $\dfrac{x^2}{a^2} + \dfrac{y^2}{b^2} = 1$

eccentricity: $e = \dfrac{1}{a}\sqrt{a^2 - b^2}, \quad a \geq b$

foci: $F_1\,(c, 0), F_2\,(-c, 0)$, where $c = \sqrt{a^2 + b^2}$

vertices: $(\pm a, 0), \ (0, \pm b)$

directrices: $D_1 - D_1, D_2 - D_2$

distance from center O to a directrix: $\dfrac{a}{e}$

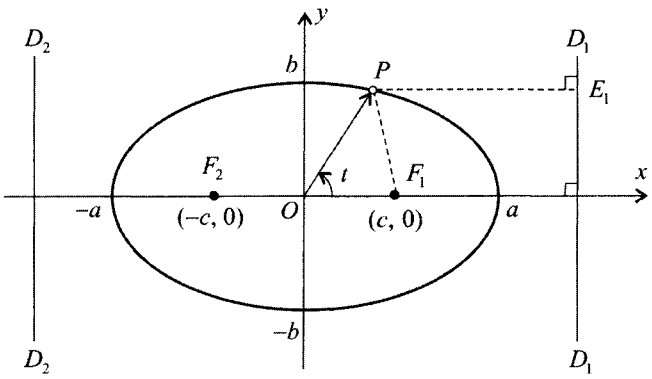


FIGURE A9

The ellipse in Fig. A9 can also be described by

$$x = a\cos t, \quad y = a\sin t$$

where t is the angle between the positive x-axis and line OP.

A circle is a special case of ellipses with $a = b$, $e = 0$, and foci coinciding at the center O.

Parabolas ($e = 1$, Fig. A10)

standard equation: $y^2 = 2px$

focus: $F\ (p/2,\ 0)$

vertex: origin $(0,\ 0)$

directrix: D-$D\ (x = -p/2)$

distance from focus to directrix: p

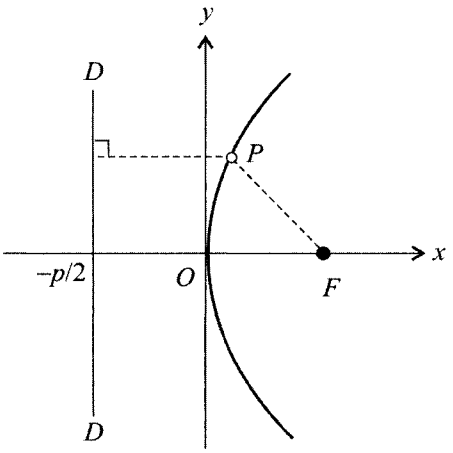


FIGURE A10

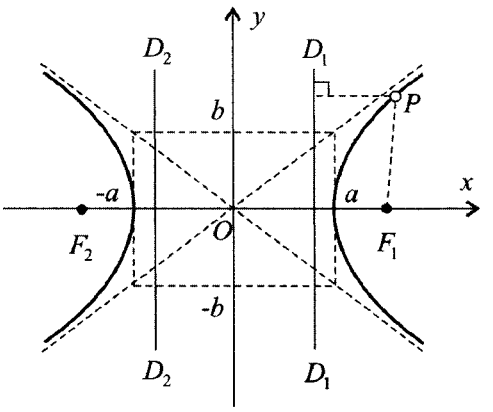


FIGURE A11

Hyperbolas ($e > 1$, Fig. A11)

standard equation: $\dfrac{x^2}{a^2} - \dfrac{y^2}{b^2} = 1$

eccentricity: $e = \dfrac{c}{a}, c = \sqrt{a^2 + b^2}$

foci: $F_1\,(c, 0), F_2\,(-c, 0)$

vertices: $(a, 0), \ (-a, 0)$

directrices: $D_1 - D_1, D_2 - D_2$

distance from center O to directrix: $\dfrac{a}{e}$

slopes of asymptotes: $\pm\dfrac{b}{a}$

Planes in Three-Dimensional Space

General form: $Ax + By + Cz + D = 0$

Intercept form: $\dfrac{x}{a} + \dfrac{y}{b} + \dfrac{z}{c} = 1$

Straight Lines in Three-Dimensional Space

General form:
$$\begin{cases} A_1x + B_1y + C_1z + D_1 = 0 \\ A_2x + B_2y + C_2z + D_2 = 0 \end{cases}$$

Point-direction form: $\dfrac{x - x_1}{a} = \dfrac{y - y_1}{b} = \dfrac{z - z_1}{c}$

A.8 Vector Analysis

Any quantity that is characterized by its magnitude and direction is called a *vector*. In mechanics of solids and structures, vectors are used to describe forces, moments, velocities, accelerations, etc.

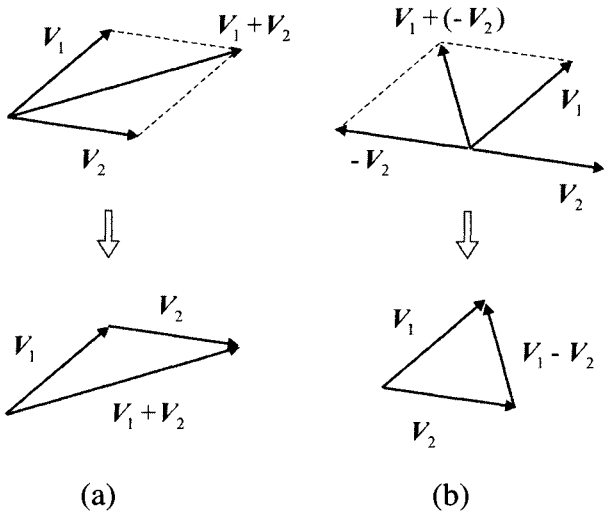


FIGURE A12

Vector Summation

The summation of two vectors, say $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$, follows the parallelogram rule as shown in Fig. A12(a). The difference of $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ is shown in Fig. A12(b), which is viewed as the sum of $\boldsymbol{V}_1$ and $-\boldsymbol{V}_2$. Here $-\boldsymbol{V}_2$ is a vector that has the same magnitude as $\boldsymbol{V}_2$, but is in the opposite direction.

Scalar Components of Vectors

Consider a rectangular frame $Oxyz$ in Fig. A13 whose unit base vectors are $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$. A vector $\boldsymbol{V}$ in the space is expressed as

$$\boldsymbol{V} = a\boldsymbol{i} + b\boldsymbol{j} + c\boldsymbol{k}$$

where a, b, and c are the projections of $\boldsymbol{V}$ along the x-, y-, and z- axes. These scalars, which are called the *components of the vector*, are given by

$$a = V\cos\theta_x, \quad b = V\cos\theta_y, \quad c = V\cos\theta_z$$

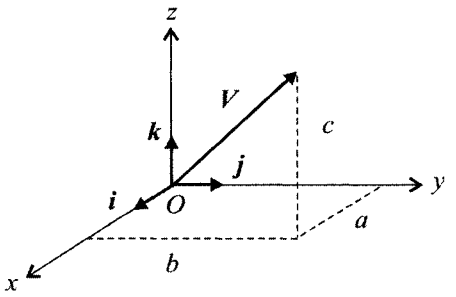


FIGURE A13

where θ_x, θ_y, θ_z are the angles between vector $\boldsymbol{V}$ and the unit vectors of the frame $Oxyz$ and

$$V = |\boldsymbol{V}| = \sqrt{a^2 + b^2 + c^2}$$

is the magnitude of $\boldsymbol{V}$. The $\cos\theta_x$, $\cos\theta_y$, and $\cos\theta_z$ are called *direction cosines.*

With vector components, a linear combination of vectors $\boldsymbol{V}_1 = x_1\boldsymbol{i} + y_1\boldsymbol{j} + z_1\boldsymbol{k}$ and $\boldsymbol{V}_2 = x_2\boldsymbol{i} + y_2\boldsymbol{j} + z_2\boldsymbol{k}$ is expressed as

$$\alpha\boldsymbol{V}_1 + \beta\boldsymbol{V}_2 = (\alpha x_1 + \beta x_2)\boldsymbol{i} + (\alpha y_1 + \beta y_2)\boldsymbol{j} + (\alpha z_1 + \beta z_2)\boldsymbol{k}$$

where α and β are two scalars.

Dot Product of Vectors

The dot (scalar) product of vectors $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ is

$$\boldsymbol{V}_1 \cdot \boldsymbol{V}_2 = \boldsymbol{V}_2 \cdot \boldsymbol{V}_1 = |\boldsymbol{V}_1| \cdot |\boldsymbol{V}_2| \cos\theta$$

where θ is the angle between the vectors; see Fig. A14. By vector components, the dot product is

$$\boldsymbol{V}_1 \cdot \boldsymbol{V}_2 = \boldsymbol{V}_2 \cdot \boldsymbol{V}_1 = x_1x_2 + y_1y_2 + z_1z_2.$$

Note that $\boldsymbol{V}_1 \cdot \boldsymbol{V}_2 = 0$ if $\boldsymbol{V}_1$ is perpendicular to $\boldsymbol{V}_2$. In particular, the unit vectors in Fig. A13 satisfy

$$\boldsymbol{i} \cdot \boldsymbol{i} = \boldsymbol{j} \cdot \boldsymbol{j} = \boldsymbol{k} \cdot \boldsymbol{k} = 1; \quad \boldsymbol{i} \cdot \boldsymbol{j} = \boldsymbol{j} \cdot \boldsymbol{k} = \boldsymbol{k} \cdot \boldsymbol{i} = 0$$

It follows that for vector $\boldsymbol{V} = a\boldsymbol{i} + b\boldsymbol{j} + c\boldsymbol{k}$

$$a = \boldsymbol{i} \cdot \boldsymbol{V}, \quad b = \boldsymbol{j} \cdot \boldsymbol{V}, \quad c = \boldsymbol{k} \cdot \boldsymbol{V}.$$

Cross Product of Vectors

The cross (vector) product of vectors $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ is

$$\boldsymbol{V}_1 \times \boldsymbol{V}_2 = |\boldsymbol{V}_1| \cdot |\boldsymbol{V}_2| \sin\theta\, \boldsymbol{n}$$

where θ is the angle *from* $\boldsymbol{V}_1$ *to* $\boldsymbol{V}_2$, and $\boldsymbol{n}$ is a unit vector in the direction determined by a right-hand screw rotated from $\boldsymbol{V}_1$ *to* $\boldsymbol{V}_2$; see Fig. A15. The vector $\boldsymbol{n}$ is perpendicular to the plane made by $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$. By vector components

$$\boldsymbol{V}_1 \times \boldsymbol{V}_2 = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} = (y_1z_2 - y_2z_1)\,\boldsymbol{i} + (z_1x_2 - z_2x_1)\boldsymbol{j} + (x_1y_2 - x_2y_1)\,\boldsymbol{k}$$

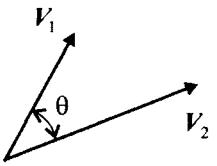


FIGURE A14

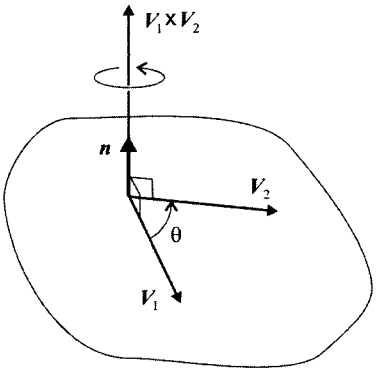


FIGURE A15

By the cross product definition,

$$V_1 \times V_2 = -V_2 \times V_1.$$

The unit vectors in Fig. A13 satisfy

$$i \times j = k, \quad j \times k = i, \quad k \times i = j$$
$$i \times i = j \times j = k \times k = 0$$

Scalar Triple Product

The scalar triple product of vectors V_1, V_2, and V_3 is

$$[V_1, V_2, V_3] = V_1 \cdot (V_2 \times V_3)$$

By vector components,

$$[V_1, V_2, V_3] = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}.$$

Also, it can be shown that

$$[V_1, V_2, V_3] = [V_2, V_3, V_1] = [V_3, V_1, V_2]$$
$$[V_2, V_1, V_3] = -[V_1, V_2, V_3].$$

Spatial Derivatives of Vectors and Scalars

Consider the rectangular frame $Oxyz$ in Fig. A13. Define the following gradient operator

$$\nabla = i\frac{\partial}{\partial x} + j\frac{\partial}{\partial y} + k\frac{\partial}{\partial z}$$

Let vector $V = V_x i + V_y j + V_z k$ be a function of the spatial coordinates x, y, and z. The divergence of V is a dot product of ∇ and the vector:

$$\text{div}V = \nabla \cdot V = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z}$$

which is a scalar function. The curl of $\boldsymbol{V}$ is a cross product of ∇ and the vector:

$$\text{curl}\boldsymbol{V} = \nabla \times \boldsymbol{V} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ \dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial z} \\ V_x & V_y & V_z \end{vmatrix}$$

In addition, for a scalar function S of x, y, and z, its gradient is defined as

$$\text{grad}\, S = \nabla S = \frac{\partial S}{\partial x}\boldsymbol{i} + \frac{\partial S}{\partial y}\boldsymbol{j} + \frac{\partial S}{\partial z}\boldsymbol{k}.$$

A.9 Matrix Theory

Definitions

An m-by-n matrix A is an array of numbers arranged in m rows and n columns as follows

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

where a_{ij} denotes the element of A in the position ij, namely, in the ith row and jth column. An m-dimensional or m-by-1 column vector x is a column of m numbers

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}$$

An n-dimensional or 1-by-n row vector y is a row of n numbers

$$y = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix}$$

An m-by-n matrix A can be viewed as an assembly of n column vectors

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}; \quad a_i = \begin{pmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{mi} \end{pmatrix}, \quad i = 1, 2, \ldots, n$$

or an assembly of m row vectors

$$A = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix}; \quad a_j = \begin{pmatrix} a_{j1} & a_{j2} & \cdots & a_{jn} \end{pmatrix}, \quad j = 1, 2, \ldots, m$$

The elements of a matrix or vector can be either real or complex.

Matrix Operations

Matrix Addition Let A and B be m-by-n matrices of elements a_{ij} and b_{ij}, respectively

$$C = A + B$$

implies that the elements of matrix C

$$c_{ij} = a_{ij} + b_{ij}.$$

Matrix Transposition The transpose of matrix A, written as A^T, is a matrix whose element in the ith row and jth column is a_{ji}. For instance

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \rightarrow \quad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}.$$

The transposition of vectors is similarly operated.

Matrix Multiplication Assume that A is an m-by-p matrix and B is a p-by-n matrix.

$$C = AB$$

implies that the elements of matrix C

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}.$$

Products of Vectors Consider vectors

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}, \quad z = \begin{pmatrix} z_1 & z_2 & \cdots & z_m \end{pmatrix}$$

The products $x^T y$ and zy are scalars

$$x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_m y_m$$
$$zy = z_1 y_1 + z_2 y_2 + \cdots + z_m y_m$$

The products xy^T and xz produce the following matrices

$$xy^T = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_m \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_m \\ \vdots & \vdots & \cdots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_m \end{bmatrix}, \quad xz = \begin{bmatrix} x_1 z_1 & x_1 z_2 & \cdots & x_1 z_m \\ x_2 z_1 & x_2 z_2 & \cdots & x_2 z_m \\ \vdots & \vdots & \cdots & \vdots \\ x_m z_1 & x_m z_2 & \cdots & x_m z_m \end{bmatrix}$$

Products of Matrices and Vectors Let A be an m-by-n matrix. Let x be an n-by-1 column vector. The product

$$y = Ax$$

is an m-by-1 column vector with the elements

$$y_k = \sum_{j=1}^{n} a_{kj}x_j, \quad k = 1, 2, \ldots, m$$

Square Matrices

A is a square matrix if it has the same number of rows and columns ($m = n$).

Some Special Square Matrices Identity matrix: a matrix with the elements in all the diagonal ii being ones and all other elements being zeros. For instance,

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Diagonal matrix: a matrix with all the elements on the off-diagonal positions (ij, $i \neq j$) being zeros. For instance,

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} = \text{diag}\begin{Bmatrix} 1 & 2 & 3 \end{Bmatrix}$$

Symmetric matrix: $A^T = A$.

Antisymmetric (skew-symmetric) matrix: $A^T = -A$.

Determinant The determinant of a 2-by-2 matrix is

$$\det A = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

The determinant of a 3-by-3 matrix is

$$\det A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

The determinant of an n-by-n matrix A of elements a_{ij} can be computed by

$$\det A = \sum_{j=1}^{n} (-1)^{j+1} a_{1j} \det A_{1j}$$

where A_{1j} is an $(n-1) \times (n-1)$ matrix obtained by deleting the first row and jth column of A.

Inversion The inverse of matrix A, written as A^{-1}, is a matrix such that

$$AA^{-1} = I, \quad A^{-1}A = I$$

where I is the identity matrix. For a 2-by-2 matrix A, its inverse is

$$A^{-1} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

The inverse of an n-by-n matrix can be computed by

$$A = \frac{\text{adj}\, A}{\det A}$$

where adj A is the adjoint of A.

The inverse A^{-1} exists if and only if $\det A \neq 0$. Matrix A is nonsingular if $\det A \neq 0$, and is singular if $\det A = 0$. So, only nonsingular matrices have inverses.

Properties of Matrix Operations

$$\det(AB) = \det A \det B$$

$$\det A^T = \det A$$

$$(AB)^T = B^T A^T$$

$$(A + B)^T = A^T + B^T$$

$$\left(A^{-1}\right)^T = \left(A^T\right)^{-1}$$

Let a and b be n-vectors. If $1 + b^T a \neq 0$, then

$$\left(I + ab^T\right)^{-1} = I - \frac{1}{1 + b^T a} ab^T$$

where I is the n-by-n identity matrix.

Linear Independence of Vectors

Consider a set of vectors $x_1, x_2, \ldots, x_n$. If

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n = 0$$

requires that coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$ are all equal to zero, the vectors are said to be *linearly independent*. Otherwise, the vectors are linearly dependent. Furthermore, if $x_1, x_2, \ldots, x_n$ are $n \times 1$ linearly independent vectors,

$$\det \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \neq 0.$$

Rank of Matrix

The rank of a matrix is defined as the number of linearly independent columns or the number of linearly independent rows, where the columns and rows of the matrix are viewed as vectors. The rank of a matrix A is denoted by rank(A). For an m-by-n matrix A

$$\text{rank}(A) \leq \min(m, n)$$

For a square matrix $A(m = n)$, rank$(A) = n$ if $\det A \neq 0$; rank$(A) < n$ if $\det A = 0$. As an example, for the matrices

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ -5 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

rank$(A) = 2$, rank$(B) = 1$, and rank$(C) = 0$.

Eigenvalues and Eigenvectors

Consider an n-by-n matrix A. The homogeneous equation

$$Av = \lambda v \quad \text{or} \quad (\lambda I - A)v = 0$$

defines an eigenvalue problem, where the unknown parameter λ is an eigenvalue of A, and the unknown vector v is an eigenvector. The matrix A has n eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ that are the roots of the characteristic equation

$$\det(\lambda I - A) = \lambda^n + \alpha_{n-1}\lambda^{n-1} + \cdots \alpha_1 \lambda + \alpha_0 = 0.$$

Associated with each eigenvalue, there is an eigenvector satisfying

$$Av_k = \lambda_k v_k, \quad k = 1, 2, \ldots, n.$$

As an example, for the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix}$$

the characteristic equation is $\det(\lambda I - A) = \lambda^2 - 7\lambda = 0$, and the eigenvalues and eigenvectors are

$$\lambda_1 = 0, \quad v_1 = \begin{pmatrix} -2 \\ 1 \end{pmatrix}; \quad \lambda_2 = 7, \quad v_2 = \begin{pmatrix} 1 \\ 3 \end{pmatrix}.$$

Exponentials of Matrices

The exponential of an n-by-n matrix A is defined as the infinite series

$$e^{At} = I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 + \cdots$$

where t is a parameter. An example is given below

$$A = \begin{bmatrix} \sigma & \omega \\ -\omega & \sigma \end{bmatrix}, \quad e^{At} = e^{\sigma t}\begin{bmatrix} \cos \omega t & \sin \omega t \\ -\sin \omega t & \cos \omega t \end{bmatrix}$$

If matrix A has n linearly independent eigenvectors $v_1, v_2, \cdots, v_n$ corresponding to its eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_n$, then

$$e^{At} = Te^{\Lambda t}T^{-1}$$

where

$$e^{\Lambda t} = \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & e^{\lambda_n t} \end{bmatrix}$$

and matrix T is composed of the eigenvectors

$$T = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}$$

Properties of Exponential Matrices

(i) $\dfrac{d}{dt}\left(e^{At}\right) = Ae^{At} = e^{At}A$

(ii) $\left(e^{At}\right)^{-1} = e^{-At}$

(iii) $e^{At_1}e^{At_2} = e^{A(t_1+t_2)}$

(iv) $e^{A\cdot 0} = I$, where I is an identity matrix.

Solution of State Equations by Exponential Matrices

Consider a system of first-order differential equations

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$x(0) = x_0$$

where $x(t)$ is a vector of n time-dependent variables, i.e.,

$$x(t) = \begin{pmatrix} x_1(t) & x_2(t) & \cdots & x_n(t) \end{pmatrix}^T;$$

$u(t)$ is a vector of m given functions of time t; A is an n-by-n constant matrix; B is an n-by-m constant matrix; and x_0 is a vector of initial values of the variables $x_1, \ldots, x_n$. The differential equations described above are normally called *state equations*, $x_1, \ldots, x_n$ *state variables*, and $u(t)$ *input vector*.

The solution of the above state equations is given by

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

where e^{At} is the exponential matrix defined previously. The exponential matrix is also referred to as the *fundamental matrix* or *transition matrix* of the system (of differential equations).

A.10 Complex Numbers and Complex Functions

A complex number z, which is illustrated in Fig. A16, is expressed by

$$z = x + jy$$

where x and y are real numbers, and j is the imaginary unit of the form $j = \sqrt{-1}$. The x and y are the *real and imaginary parts* of z, respectively. In the polar coordinates,

$$z = re^{j\theta} = r(\cos\theta + j\sin\theta)$$

Modulus $\quad |z| = r = \sqrt{x^2 + y^2}$

Argument $\quad \arg z = \theta = \arctan(y/x)$

Complex Conjugate

For a complex number $z = x + jy$, its complex conjugate is

$$\bar{z} = x - jy$$

It can be shown that

$$|\bar{z}| = |z| = \sqrt{x^2 + y^2}, \quad \arg\bar{z} = -\arg z$$

$$z\bar{z} = |z|^2 = x^2 + y^2, \quad z + \bar{z} = 2x, \quad z - \bar{z} = 2yj$$

Algebraic Manipulations

$$(a + jb) \pm (c + jd) = (a \pm c) + j(b \pm d)$$

$$(a + jb) \times (c + jd) = (ac - bd) + j(bc + ad)$$

$$\frac{a + jb}{c + jd} = \frac{1}{c^2 + d^2}\{(ac + bd) + j(bc - ad)\}$$

Trigonometric Manipulations

For $z_1 = r_1 e^{j\theta_1}$ and $z_2 = r_2 e^{j\theta_2}$,

$$z_1 z_2 = re^{j\theta}, \quad \text{with } r = r_1 r_2, \quad \theta = \theta_1 + \theta_2$$

$$\frac{z_1}{z_2} = \rho e^{j\beta}, \quad \text{with } \rho = \frac{r_1}{r_2}, \quad \beta = \theta_1 - \theta_2$$

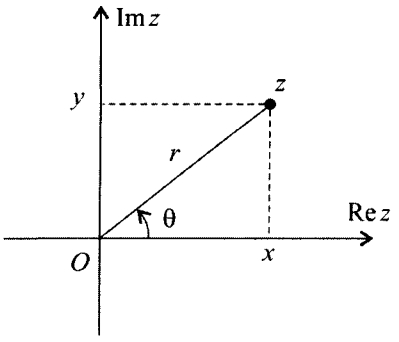


FIGURE A16

Functions of Complex Variable

A function of complex variable $s = x + jy$ can be written as

$$f(s) = u(x, y) + jv(x, y)$$

where u and v are the real part and imaginary part of the function. The function is said to be analytic at point s_0 if it is differentiable in a neighborhood of s_0 in the complex plane.

Cauchy-Riemann Conditions The necessary and sufficient conditions that $f(s)$ is analytic are

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$$

Taylor Series If $f(s)$ is analytic in the circle $C = \{s : |s - s_0| \leq r_0\}$, then it can be expressed as a convergent series

$$f(s) = \sum_{k=0}^{\infty} a_k (s - s_0)^k, \quad |s - s_0| \leq r_0$$

with

$$a_k = \frac{1}{2\pi j} \int_C \frac{f(s)}{(s - s_0)^{k+1}} ds$$

Rational Functions and Its Poles and Zeros

A rational function of s is defined as

$$F(s) = \frac{B(s)}{A(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}$$

where $A(s)$ and B(s) are polynomials of s. The poles of $F(s)$ are the roots of the denominator polynomial equation

$$A(s) = 0$$

The zeros of $F(s)$ are the roots of the numerator polynomial equation

$$B(s) = 0$$

For example, function $F(s) = \frac{s+3}{s(s^2+2s+5)}$ has three poles $0, -1 \pm j2$, and one zero -3.

A.11 Laplace Transforms

Definition

For a real-valued, piecewise continuous function $f(t)$ specified for $t \geq 0$, its Laplace transform, denoted by $F(s)$, is defined by

$$F(s) = \mathscr{L}\{f(t)\} = \int_0^{\infty} e^{-st} f(t) dt$$

where $\mathscr{L}$ is the Laplace transform operator, and s is the complex Laplace transform parameter whose real part is greater than some fixed number σ_0. Furthermore, $f(t)$ can be expressed by

the inverse Laplace transform of $F(s)$:

$$f(t) = \mathcal{L}^{-1}\{F(s)\} = \frac{1}{2\pi i}\int_{\sigma-i\infty}^{\sigma+i\infty} e^{st}F(s)ds$$

where $\mathcal{L}^{-1}$ is the inverse Laplace transform operator, $\sigma > \sigma_0$ and $i = \sqrt{-1}$. The $f(t)$ and $F(s)$ are a Laplace transform pair.

Properties of Laplace Transforms

1. Superposition or linear combination

$$\mathcal{L}\{\alpha f_1(t) + \beta f_2(t)\} = \alpha\mathcal{L}\{f_1(t)\} + \beta\mathcal{L}\{f_2(t)\} = \alpha F_1(s) + \beta F_2(s)$$

2. Time delay or shift in time
   Given a function $f(t)$, define a time-shifted function $f_d(t-T)$ by

$$f_d(t-T) = \begin{cases} f(t-T), & t > T \\ 0, & t < T \end{cases}$$

   where T is a positive time delay parameter. Then

$$\mathcal{L}\{f_d(t-T)\} = e^{-sT}\mathcal{L}\{f(t)\} = e^{-sT}F(s).$$

3. Differentiation

$$\mathcal{L}\left\{\frac{df(t)}{dt}\right\} = sF(s) - f(0)$$

$$\mathcal{L}\left\{\frac{d^2f(t)}{dt^2}\right\} = s^2F(s) - sf(0) - \frac{d}{dt}f(0)$$

$$\mathcal{L}\left\{\frac{d^nf(t)}{dt^n}\right\} = s^nF(s) - s^{n-1}f(0) - s^{n-2}f^{(1)}(0) - \cdots - f^{(n-1)}(0)$$

   where $f^{(k)} = \frac{d^kf}{dt^k}$.

4. Integration

$$\mathcal{L}\left\{\int_0^t f(\tau)d\tau\right\} = \frac{1}{s}F(s).$$

5. Initial value theorem
   For a Laplace transform pair $f(t)$ and $F(s)$

$$\lim_{t\to 0} f(0) = \lim_{s\to\infty} sF(s)$$

   if the indicated limits exist.

6. Final value theorem
   Let $f(t)$ and $F(s)$ be a Laplace transform pair. If $sF(s)$ is analytic on the imaginary and in the right-half of the s-plane (namely, all the poles of $sF(s)$ have negative real parts), then

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s).$$

TABLE A5 Laplace Transform Pairs

$f(t)$	$F(s)$
$\delta(t)$ (unit impulse)	1
1 (unit step)	$\frac{1}{s}$
t	$\frac{1}{s^2}$
t^n	$\frac{n!}{s^{n+1}}$
e^{-at}	$\frac{1}{s+a}$
$t^n e^{-at}$	$\frac{n!}{(s+a)^{n+1}}$
$\sin \omega t$	$\frac{\omega}{s^2+\omega^2}$
$\cos \omega t$	$\frac{s}{s^2+\omega^2}$
$e^{-at} \sin \omega t$	$\frac{\omega}{(s+a)^2+\omega^2}$
$e^{-at} \cos \omega t$	$\frac{s+a}{(s+a)^2+\omega^2}$
$\frac{\omega_n}{\sqrt{1-\xi^2}} e^{-\xi\omega_n t} \sin\left(\omega_n\sqrt{1-\xi^2}\, t\right)$	$\frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$
$e^{-\xi\omega_n t}\left\{\cos\left(\omega_n\sqrt{1-\xi^2}\, t\right) - \frac{\xi}{\sqrt{1-\xi^2}} \sin\left(\omega_n\sqrt{1-\xi^2}\, t\right)\right\}$	$\frac{s}{s^2+2\xi\omega_n s+\omega_n^2}$
$1 - e^{-\xi\omega_n t}\left\{\cos\left(\omega_n\sqrt{1-\xi^2}\, t\right) + \frac{\xi}{\sqrt{1-\xi^2}} \sin\left(\omega_n\sqrt{1-\xi^2}\, t\right)\right\}$	$\frac{\omega_n^2}{s\left(s^2+2\xi\omega_n s+\omega_n^2\right)}$
$t \sin \omega t$	$\frac{2\omega s}{\left(s^2+\omega^2\right)^2}$
$t \cos \omega t$	$\frac{s^2-\omega^2}{\left(s^2+\omega^2\right)^2}$

7. Convolution theorem
   If $F_1(s) = \mathscr{L}\{f_1(t)\}$ and $F_2(s) = \mathscr{L}\{f_2(t)\}$, then

$$\mathscr{L}^{-1}\{F_1(s)F_2(s)\} = \int_0^t f_1(\tau) f_2(t-\tau)\, d\tau = f_1 * f_2$$
$$= \int_0^t f_2(\tau) f_1(t-\tau)\, d\tau = f_2 * f_1$$

   where $f_1(t-\tau)$ and $f_2(t-\tau)$ are both time-shifted according to $f_1(t)$ and $f_2(t)$, respectively.

Table A5 lists some basic functions and their Laplace transforms.

A.12 Inverse Laplace Transform via Partial Fraction Expansion

In engineering analyses, inverse Laplace transform is frequently performed to derive analytical solutions. Partial fraction expansion is a commonly used technique for conducting inverse Laplace transform of rational functions. In this section, inverse Laplace transform by partial fraction expansion is reviewed, and several related MATLAB functions are introduced. (For the utility of MATLAB, refer to Appendix B.)

Consider a rational function of the form

$$F(s) = \frac{N(s)}{D(s)} = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}; \quad m \le n, \quad a_n \neq 0 \tag{A.1}$$

where all the coefficients are real. When $m = n$, by long division, Eq. (A.1) can be written as

$$F(s) = k + \frac{\hat{N}(s)}{D(s)} = k + \frac{\hat{b}_{n-1}s^{n-1} + \cdots + \hat{b}_1 s + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} \tag{A.2}$$

where the direct term coefficient $k = b_n/a_n$, and

$$\hat{b}_j = b_j - a_j\frac{b_n}{a_n} = b_j - ka_j, \quad j = 1, 2, \ldots, n-1 \tag{A.3}$$

As can be seen, Eq. (A.2) is also valid for $m < n$, for which $k = 0$ and $\hat{N}(s) = N(s)$. The denominator $D(s)$ has n roots, p_j, $j = 1, 2, \ldots, n$; namely, $D(s) = a_n(s-p_1)(s-p_2)\cdots(s-p_n)$. These roots are also called the *poles* of the function $F(s)$.

In what follows, partial fraction expression and inverse Laplace transform are presented in two cases of the poles of $F(s)$.

Case 1. All poles of $F(s)$ are distinct

The partial fraction expansion of $F(s)$ is

$$F(s) = k + \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \cdots + \frac{r_n}{s-p_n} \tag{A.4}$$

where r_j is the residue of $F(s)$ at pole p_j and is given by

$$r_j = \lim_{s \to p_j}(s-p_j)\frac{\hat{N}(s)}{D(s)} \tag{A.5}$$

By Eq. (A.4), the inverse Laplace transform is

$$f(t) = \mathcal{L}^{-1}[F(s)] = k\delta(t) + r_1 e^{p_1 t} + r_2 e^{p_2 t} + \cdots + r_n e^{p_n t} \tag{A.6}$$

where $\delta(t)$ is the Dirac delta function.

Case 2. Poles of $F(s)$ are repeated

Assume that $F(s)$ has l different poles:

$$D(s) = a_n(s-p_1)^{m_1}(s-p_2)^{m_2}\cdots(s-p_l)^{m_l} \tag{A.7}$$

where $m_1 + m_2 + \cdots + m_l = n$. The integer m_j (≥ 1) is the multiplicity of pole p_j. The partial fraction expansion of $F(s)$ is of the form

$$\begin{aligned} F(s) = k &+ \frac{r_{1,1}}{s-p_1} + \frac{r_{1,2}}{(s-p_1)} + \cdots + \frac{r_{1,m_1}}{(s-p_1)^{m_1}} \\ &+ \frac{r_{2,1}}{s-p_2} + \frac{r_{2,2}}{(s-p_2)} + \cdots + \frac{r_{2,m_2}}{(s-p_2)^{m_2}} \\ &+ \cdots + \frac{r_{l,1}}{s-p_l} + \frac{r_{l,2}}{(s-p_l)} + \cdots + \frac{r_{l,m_l}}{(s-p_l)^{m_l}} \end{aligned} \tag{A.8}$$

where the residues about the pole are given by

$$r_{j,m_j-i} = \frac{1}{i!} \lim_{s \to p_j} \frac{d^i \Delta(s)}{ds^i}, \quad i = 0, 1, \ldots, m_j - 1 \tag{A.9}$$

with

$$\Delta(s) = \frac{\hat{N}(s)}{D(s)}(s-p_j)^{m_j} = \frac{\hat{N}(s)}{a_n \prod\limits_{\substack{i=1 \\ i \neq j}}^{l} (s-p_i)^{m_i}} \tag{A.10}$$

By Eq. (A.8), the inverse Laplace transform of $F(s)$ is

$$\begin{aligned} f(t) = \mathscr{L}^{-1}[F(s)] = k\delta(t) &+ e^{p_1 t}\left(r_{1,1} + r_{1,2}t + \frac{1}{2!}r_{1,3}t^2 + \cdots + \frac{1}{(m_1-1)!}r_{1,m_1}t^{m_1-1}\right) \\ &+ e^{p_2 t}\left(r_{2,1} + r_{2,2}t + \frac{1}{2!}r_{2,3}t^2 + \cdots + \frac{1}{(m_2-1)!}r_{2,m_2}t^{m_2-1}\right) + \cdots \\ &+ e^{p_l t}\left(r_{l,1} + r_{l,2}t + \frac{1}{2!}r_{l,3}t^2 + \cdots + \frac{1}{(m_l-1)!}r_{l,m_l}t^{m_l-1}\right) \end{aligned} \tag{A.11}$$

In many engineering problems, the poles and residues of a rational function $F(s)$ are complex, which yields complex terms in Eqs. (A.6) and (A.11). However, since $F(s)$ in Eq. (A.1) only contains real coefficients (a_i and b_i), its inverse Laplace transform must be real-valued. In what follows, we derive the real expressions for the inverse Laplace transform of $F(s)$. If p is a complex pole of multiplicity one and r is the residue of $F(s)$ at the pole, the terms $re^{pt} + \bar{r}e^{\bar{p}t}$ will be included in Eq. (A.6), where $\bar{p}$ and $\bar{r}$ are the complex conjugates of p and r, respectively. Write $p = \alpha + j\beta$, $r = \sigma + j\omega$, with $j = \sqrt{-1}$. It follows that

$$re^{pt} + \bar{r}e^{\bar{p}t} = 2\,\mathrm{Re}\left((\alpha + j\beta)\,e^{(\sigma + j\omega)t}\right) = 2e^{\sigma t}\,(\alpha \cos \omega t - \beta \sin \omega t) \tag{A.12}$$

If p is a complex pole of multiplicity m and $r_1, r_2, \ldots, r_m$ are the residues at the pole, Eq. (A.11) will contain the following terms

$$\begin{aligned} &e^{pt}\left(r_1 + r_2 t + \frac{1}{2!}r_3 t^2 + \cdots + \frac{1}{(m-1)!}r_m t^{m-1}\right) \\ &+ e^{\bar{p}t}\left(\bar{r}_1 + \bar{r}_2 t + \frac{1}{2!}\bar{r}_3 t^2 + \cdots + \frac{1}{(m-1)!}\bar{r}_m t^{m-1}\right) \end{aligned} \tag{A.13}$$

Write $p = \sigma + j\omega$, $r_i = \alpha_i + j\beta_i$, where $j = \sqrt{-1}$ and $i = 1, 2, \ldots, m$. The expression (A.13) is written as

$$\begin{aligned} &2\,\mathrm{Re}\left\{e^{pt}\left(r_1 + r_2 t + \frac{1}{2!}r_3 t^2 + \cdots + \frac{1}{(m-1)!}r_m t^{m-1}\right)\right\} \\ &= 2e^{\sigma t}\sum_{i=1}^{m}\frac{t^{i-1}}{(i-1)!}(\alpha_i \cos\omega t - \beta_i \sin\omega t) \end{aligned} \tag{A.14}$$

According to the above discussion, the inverse Laplace transform of $F(s)$ contains the following six types of real terms:

(i) $\dfrac{a}{s} \rightarrow a$

(ii) $\dfrac{a}{s^m} \rightarrow \dfrac{a}{(m-1)!}t^{m-1}$

(iii) $\dfrac{a}{s-\sigma} \rightarrow ae^{\sigma t}$

(iv) $\dfrac{a}{(s-\sigma)^m} \rightarrow \dfrac{a}{(m-1)!}t^{m-1}e^{\sigma t}$

(v) $\dfrac{r}{s-p} + \dfrac{\bar{r}}{s-\bar{p}} \rightarrow 2e^{\sigma t}(\alpha \cos\omega t - \beta \sin\omega t)$

(vi) $\dfrac{r}{(s-p)^m} + \dfrac{\bar{r}}{(s-\bar{p})^m} \rightarrow \dfrac{2}{(m-1)!}t^{m-1}e^{\sigma t}(\beta \cos\omega t - \beta \sin\omega t)$

where in types (i) to (iv), a and σ are real; in types (v) and (vi), $p = \alpha + j\beta$ and $r = \sigma + j\omega$ with $j = \sqrt{-1}$.

In summary, the inverse Laplace transform of a rational function via partial fraction expansion takes the following steps:

Step 1. For a given function $F(s)$, determine its poles p_j and multiplicities m_j;

Step 2. Determine the residues about the poles by Eq. (A.5) or (A.9);

Step 3. Obtain inverse Laplace transform of $F(s)$ by Eqs. (A.6) and (A.11); and

Step 4. Convert complex terms in Eqs. (A.6) and (A.11) to real terms by Eqs. (A.12) and (A.14).

Solution by the Toolbox

This Apppendix offers MATLAB functions `pfe` and `invLT` for obtaining partial fraction expansion and inverse Laplace transform of rational functions; see Windows A.1 and A.2.

Window A.1. Function pfe

MATLAB Function: **pfe**

Purpose:
To obtain partial fraction expansion of a rational function of the form

$$F(s) = \frac{N(s)}{D(s)} = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}, \quad m \le n.$$

Window A.1. Function pfe *(Continued)*

Synopsis:

```
pfe(num,den)
[r,p,k] = pfe(num,den)
```

Description:

`pfe(num,den)` finds the residues, poles, and direct term coefficient of a partial fraction expansion of the function $F(s)$ according to Eqs. (A.4) and (A.8) and displays the results. The row vectors `num` and `den` specify the coefficients of the numerator $N(s)$ and the denominator $D(s)$ in descending powers of s; namely, `num` $= [b_m \quad b_{m-1} \quad \cdots \quad b_1 \quad b_0]$ and `den` $= [a_n \quad a_{n-1} \quad \cdots \quad a_1 \quad a_0]$. The order of the numerator should not be larger than that of the denominator; i.e., $m \leq n$.

`[r,p,k] = pfe(num,den)` returns the residues of the partial fraction expansion in a vector `r`, the poles of $F(s)$ in a vector `p`, and the direct term coefficient in a scalar `k`.

Window A.2. Function invLT

MATLAB Function: **invLT**

Purpose:

To obtain the inverse Laplace transform of a rational function

$$F(s) = \frac{N(s)}{D(s)} = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}, \quad m \leq n$$

by partial fraction expansion.

Synopsis:

```
[y, k] = invLT(num, den)
```

Description:

`[y, k] = invLT(num, den)` finds the inverse Laplace transform of function $F(s)$, where the row vectors `num` and `den` specify the coefficients of the numerator $N(s)$ and the denominator $D(s)$ (see Window A.1), `y` is a matrix specifying the mathematical expression of the inverse-Laplace-transformed function, and `k` is the direct term coefficient from the partial fraction expansion of $F(s)$. The matrix `y` is of the form

$$\mathtt{y} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{n_y} \end{bmatrix} \in R^{n_y \times 6}$$

where n_y is the number of terms from the inverse Laplace transform and the 1-by-6 vector α_l has the format given in Table A.6. The direct term coefficient `k` $= 0$ when $m < n$; `k` $= b_n/a_n$ when $m = n$.

TABLE A6 The Format of Vectors α_l

Vector α_l	*Mathematical Expression*
[0 0 0 0 0 0]	$z(t) = 0$
[1 a 0 0 0 0]	$z(t) = a$
[2 a m 0 0 0]	$z(t) = at^m$
[3 a σ 0 0 0]	$z(t) = ae^{\sigma t}$
[4 a σ m 0 0]	$z(t) = at^m e^{\sigma t}$
[5 a b σ ω 0]	$z(t) = e^{\sigma t}(a\cos\omega t + b\sin\omega t)$
[6 a b σ ω m]	$z(t) = t^m e^{\sigma t}(a\cos\omega t + b\sin\omega t)$

Note. The α_l mentioned in Window A.2 are 1-by-6 vectors with the format given in Table A.6. For instance, the matrix

$$y = \begin{bmatrix} 1 & -0.25 & 0 & 0 & 0 & 0 \\ 5 & 0.264 & -0.178 & -0.359 & 6.785 & 0 \\ 4 & 0.867 & -0.051 & 1 & 0 & 0 \end{bmatrix}$$

indicates that

$$\mathscr{L}^{-1}[F(s)] = -0.25 + e^{-0.359t}(0.264\cos 6.785t - 0.178\sin 6.785t) + 0.867te^{-0.051t}.$$

The inverse Laplace transform obtained by function `invLT` can be plotted against time by function `plotLT`; see Window A.3.

Window A.3. Function `plotLT`

MATLAB Function: **`plotLT`**

Purpose:
To plot an inverse-Laplace-transformed function versus time.

Synopsis:

```
plotLT(ya, tf, npts, Td)
[y, t] = plotLT(ya, tf, npts, Td)
```

Description:
`plotLT(ya, tf, npts, Td)` computes and plots the time history of a function $f(t)$ that is obtained by $f(t) = \mathscr{L}^{-1}[F(s)]$, with $F(s)$ being a rational function of s given by Eq. (A.1). Here `ya` is a matrix obtained by function `invLT`, i.e.,

```
ya = invLT(num, den)
```

that specifies function $f(t)$ according to Table A6; `tf` is total computation time; `npts` is the number of time points used in computation; and `Td` is time shift (time delay) of the time history of $f(t)$. By default, `tf` $= 4T$, where T is a time constant that is determined by the poles of $F(s)$; `npts` $= 501$; and `Td` $= 0$. Any one of the parameters `tf`, `npts`, and `Td` can be replaced by [ ] as a placeholder for its default value.

`[y, t] = plotLT(ya, tf, npts, Td)` returns the computed values of $f(t)$ in a vector `y` and the times of computation in a vector `t`. With `y` and `t`, the time history of $f(t)$ can be plotted by the MATLAB function `plot(t, y)`.

EXAMPLE A.1

Consider the function

$$F(s) = \frac{2s^2 + s + 6}{s^3 + 5s^2 + 17s + 13}$$

```
>> [r, p, k] = pfe([2 1 6], [1 5 17 13])
```

yields

```
r = (0.7000 0.6500 + 0.9500i 0.6500 - 0.9500i)^T
p = ( - 1.0000 - 2.0000 + 3.0000i - 2.0000 - 3.0000i)^T
k = 0
```

which gives the partial fraction expansion of the function

$$F(s) = \frac{0.7}{s+1} + \frac{0.65 + i0.95}{s+2-i3} + \frac{0.65 - i0.95}{s+2+i3}.$$

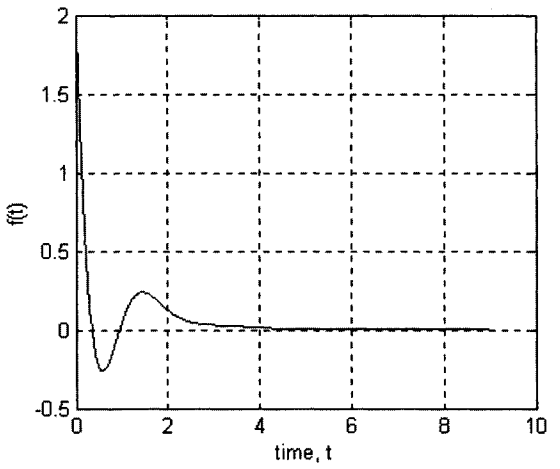


FIGURE A17

Furthermore, by

```
>> ya = invLT([2 1 6], [1 5 17 13]);
>> plotLT(ya)
```

the inverse Laplace transform of $F(s)$ is obtained as follows

$$f(t) = 0.7e^{-t} + e^{-2t}(1.3\cos 3t - 1.9\sin 3t)$$

and $f(t)$ is plotted in Fig. A17.

Appendix B MATLAB Basics

Inside
- Getting Started
- Matrix and Vector Manipulations
- Graphics
- M-Files
- Control Flow
- Solution of Algebraic and Differential Equations
- Control System Toolbox

MATLAB is a premier software package that provides an interactive environment for technical computation, data analysis, graphics, and visualization. MATLAB has many predefined functions (subroutines) that make numerical solution of engineering problems and computer programming much easier, compared to computing via other languages.

B.1 Getting Started

After MATLAB is launched, a prompt >> appears in the command window. Right next to the prompt is a cursor blinking, indicating that the system is ready to receive commands.

Assignment of Variables

In MATLAB, all variable names (scalars, vectors, and matrices) have to be assigned numerical values prior to being used in an expression. The syntax of variable assignment is

variable name = *an expression* or *a value*

If the = sign and variable name are omitted in a statement, a name `ans` is automatically assigned by MATLAB. For instance, typing the following command (and hitting the Enter or Return key)

```
>> b=-1.5
```

yields the output (displayed in the command window)

```
b=
-1.5000
```

which defines a scalar b with value -1.5. But

```
>> -1.5
ans =
-1.5000
```

The statement

```
>> v=[1; -5; 13; 8]
```

or

```
>> v=[1 -5 13 8]'
```

defines a column vector

```
v=
 1
 -5
 13
 8
```

and the statement

```
>> w=[1 -5 13 8]
```

defines a row vector

```
w=
  1 -5 13 8
```

Furthermore,

```
>> A=[1 3 5; 6 9 8; -2 0 8]
```

produces a matrix

```
A=
 1  3  5
 6  9  8
-2  0  8
```

The MATLAB Workspace

All variables assigned in the command window are stored in a part of computer memory called the *MATLAB workspace*, and these variables can be recalled at any time as desired. A list of

the variables ever assigned can be obtained by the MATLAB command who:

```
>> who
Your variables are:
A  ans  b  v  w
```

The variables that reside in the MATLAB workspace can be saved in a binary data file for later use. This is done in one of the following two ways:

- Choose the **Save Workspace As ...** from the **File** menu
- Use the MATLAB save command

```
>> save dataFile
```

which stores all the variables in the file dataFile.mat. To retrieve the saved data, either choose the **Import Data ...** from the **File** menu, or use the MATLAB load command

```
>> load dataFile
```

Character Strings

In MATLAB, single quotations (' ') are used to define and store a string of characters as an array ASCII values. For example, the statement

```
>> Book = 'Stress, Strain, and Structural Dynamics'
Book =
Stress, Strain, and Structural Dynamics
```

defines variable Book as a representation of the character string "Mechanics of Solids, and Structures." The variable Book, when typed in the command window will display its content. The character string can also be shown by

```
>> disp(Book)
Stress, Strain, and Structural Dynamics
```

or

```
>> disp('Stress, Strain, and Structural Dynamics')
Stress, Strain, and Structural Dynamics
```

Here disp is a MATLAB function that displays results (characters or numbers) without identifying variable names.

Suppression of Output Display

If a statement is followed by a semicolon (;), the display of the output will be suppressed.

```
>> A=[1 3 5; 6 9 8; -2 0 8];
```

assigns numerical values to matrix *A*, but does not show the result.

Entering Several Statements on One Line

When separated by commas (,) or semicolons (;), several commands or statements can be placed on one line. Compare the outputs of the following statements

```
>> a = 5, x = 2, y = 8
>> a = 5; x = 2; y = 8;
```

Mathematical Operators

MATLAB uses the following operators in mathematical expressions:

+	addition
−	subtraction
*	multiplication
/	division
\	left division
^	power

For instance, `a*(3.5)^3+b/c` represents the mathematical expression $a(3.5)^3 + b/c$.

Predefined Variables

MATLAB has several predefined variables, some of which are shown below:

i, j	imaginary unit $\sqrt{-1}$
pi	$\pi = 3.14159\ldots$
Inf	infinity (∞)
NaN	not a number

```
>> z=2-5*j;
```

assigns a complex number $z = 2 - j5, j = \sqrt{-1}$. These predefined variables, however, can be overwritten for other usage. For example,

```
>> j=[1 2; 3 4];
```

assigns `j` as a 2-by-2 matrix.

In addition, the letter e is used to express powers of base 10. To assign a variable `d` with value -2.5×10^3,

```
>> d=-2.5e3
d =
   -2500
```

Mathematical Functions

MATLAB offers many predefined mathematical functions for technical computations. Table B1 lists some commonly used functions, where variables x and y can be numbers, vectors or matrices. As an example, the value of the expression $y = e^{-a}\sin(x) + 10\sqrt{y}$ for $a = 5, x = 2$, and $y = 8$ is computed by

```
>> a = 5; x = 2; y = 8;
>> y = exp(-a)*sin(x) + 10*sqrt(y)
y =
 28.2904
```

Online Help

Typing "help FUN" will display in the MATLAB command window what the function FUN is and how to use it. For instance,

```
>> help rem
```

displays the information about function `rem`. Also, information about MATLAB functions can be accessed by choosing **MATLAB Help** from the **Help** menu.

TABLE B1 Elementary Functions

Function	*Meaning*	*Function*	*Meaning*
`sin(x)`	Sine	`log(x)`	Natural logarithm
`cos(x)`	Cosine	`log10(x)`	Common logarithm
`tan(x)`	Tangent	`abs(x)`	Absolute value
`asin(x)`	Arcsine	`sign(x)`	Signum function
`acos(x)`	Arcosine	`max(x)`	Maximum value
`atan(x)`	Arctangent	`min(x)`	Minimum value
`sinh(x)`	Hyperbolic sine	`ceil(x)`	Round towards $+\infty$
`cosh(x)`	Hyperbolic cosine	`floor(x)`	Round towards $-\infty$
`tang(x)`	Hyperbolic tangent	`round(x)`	Round to nearest integer
`sinh(x)`	Hyperbolic sine	`rem(x,y)`	Remainder after division
`cosh(x)`	Hyperbolic cosine	`real(x)`	Complex real part
`tang(x)`	Hyperbolic tangent	`imag(x)`	Complex imaginary part
`exp(x)`	Exponential	`angle(x)`	Phase angle
`sqrt(x)`	Square root	`conj(x)`	Complex conjugate

Number Display Formats

While computations in MATLAB are carried out in double precision, the numerical results displayed in the command window may only have four digits after the decimal point. The user can change the display format by choosing the **`Preferences ...`** from the **`File`** menu or using the following `format` commands

`format`	Default. Scaled fixed point format with 5 digits.
`format long`	Scaled fixed point format with 15 digits.
`format short e`	Floating point format with 5 digits.
`format long e`	Floating point format with 15 digits.

An example is shown below:

```
>> 1/7
ans =
  0.1429
>> format long; 1/7
ans =
  0.14285714285714
```

B.2 Matrix and Vector Manipulations

Matrix Operations

Most matrix operations in MATLAB follow the way by which conventional mathematical expressions are used in textbooks and technical references.

Transposition The transpose of a real matrix or vector is obtained by the prime or apostrophe (’):

```
>> A = [1 2; 3 4];
>> B=A'
B =
```

```
1   3
2   4
```

The transpose of a complex matrix or vector is obtained by the dot-prime (.'):

```
>> C = [1 2-j; 3+j*5 4];
>> D=C.'
D =
  1.0000            3.0000 + 5.0000i
  2.0000 - 1.0000i  4.0000
```

Note: Use of the prime alone, `C'`, produces the complex conjugate transpose of `C`.

Addition and Subtraction The symbols + and − are used for adding and subtracting matrices.

```
>> C=A+B
C =
 2   5
 5   8
```

Multiplication The symbol * is used for matrix multiplication. For instance,

```
>> C*A
ans =
 17  24
 29  42
```

and

```
>> D = 10*C
D =
 20  50
 50  80
```

Incremental Generation of Vectors

Let n_1 and n_2 be two real numbers, $n_1 \leq n_2$. The statement

$$v = n_1 : n_2$$

generates a row vector v containing the number from n_1 to n_2 with unit increment. The statement

$$w = n_1 : \Delta n : n_2$$

creates a row vector w containing the numbers from n_1 to n_2 with increment Δn. For instance,

```
>> x=-1:4
x =
  -1   0   1   2   3   4
>> y=0:pi/3:pi
y =
    0  1.0472  2.0944  3.1416
```

Vector y can also be generated by the MATLAB function `linspace` as follows

```
>> y = linspace(0, pi, 4)
```

Array Addressing

Consider the matrix

```
A =
    1    3    5
    6    9    8
   -2    0    8
```

The statement

```
>> A(1:2,:)
```

specifies the first two rows of A

```
ans =
    1    3    5
    6    9    8
```

The statement

```
>> A(2:3, [1 3])
```

specifies a submatrix that is obtained by deleting the first row and second column of A:

```
ans =
    6    8
   -2    8
```

In MATLAB, individual matrix elements are referenced by their subscripts. For instance, A(2,3) refers to the element of A in the second row and third column.

```
>> A(2,3)=A(3,1)-7
```

results in

```
A =
    1    3    5
    6    9   -9
   -2    0    8
```

The above array addressing is also applicable to vectors.

Element-by-Element Array Operations

Besides the standard matrix/vector operations, MATLAB performs element-by-element array operations (multiplication, division, and power) among matrices or vectors of the same dimensions. To illustrate this special feature, consider two vectors of n elements

$$a = [a_1 \quad a_2 \quad \cdots \quad a_n], \quad b = [b_1 \quad b_2 \quad \cdots \quad b_n]$$

Let g be a scalar. Some basic element-by-element array operations are given as follows:

Array multiplication	$a.*b = [a_1 * b_1 \quad a_2 * b_2 \quad \cdots \quad a_n * b_n]$
Array division	$a./b = [a_1/b_1 \quad a_2/b_2 \quad \cdots \quad a_n/b_n]$
Array powers	$a.\hat{}b = [a_1\hat{}b_1 \quad a_2\hat{}b_2 \quad \cdots \quad a_n\hat{}b_n]$
Scalar addition	$a+g = [a_1+g \quad a_2+g \quad \cdots \quad a_n+g]$

Here, operators .* (dot multiplication), ./ (dot division), and .^ (dot power) are used for element-by-element array operations. These operators make programming for computation compact and efficient. For example, to evaluate the values of the function $y = 1 - t^2 e^{-2t} \sin 5t$ at $t = 0, 0.1, 0.2, \ldots, 0.8$, write

```
>> t=0:0.1:0.8;
>> y=1-t.^2.*exp(-2*t).*sin(5*t)
y =
  1.0000  0.9961  0.9774  0.9507  0.9346  0.9450  0.9847  1.0424  1.0978
```

The above-mentioned element-by-element operations are also applicable to matrices of the same dimensions. For the two matrices

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

```
>> A./B
ans =
  0.2000  0.3333
  0.4286  0.5000
```

and

```
>> A-1
ans =
   0   1
   2   3
```

Matrix Functions

MATLAB provides many matrix functions for various matrix/vector manipulations; see Table B2 for some of these functions. Use the online help of MATLAB to find how to use these functions. As an example, consider the following matrix

```
>> A = [1 0 5; 2 -3 1; 4 4 7];
```

The inverse of matrix A is obtained as

```
>> B=inv(A)
B =
 -0.3333   0.2667   0.2000
 -0.1333  -0.1733   0.1200
  0.2667  -0.0533  -0.0400
```

TABLE B2 Matrix Functions

Function	*Utility*
`det`	Determinant
`diag`	Diagonal matrices and diagonals of a matrix
`eig`	Eigenvalues and eigenvectors
`expm`	Matrix exponentials
`eye`	Identity matrix
`inv`	Matrix inverse
`norm`	Matrix and vector norms
`ones`	Matrix containing all ones
`rank`	Number of linearly independent rows or columns
`zeros`	Matrix containing all zero elements

The determinant of matrix A is

```
>> det(A)
ans =
  75
```

The eigenvalues and eigenvectors of matrix A are given by

```
>> [U,V]=eig(A)
U =
  0.4847        0.6331             0.6331
  0.1420        0.3504 - 0.5102i   0.3504 + 0.5102i
  0.8631       -0.4371 + 0.1581i  -0.4371 - 0.1581i
V =
  9.9042             0                  0
     0       -2.4521 + 1.2489i          0
     0                0        -2.4521 - 1.2489i
```

where the diagonal elements of matrix V are the eigenvalues and the columns of matrix U are the associate eigenvectors.

Polynomial Manipulations

In MATLAB, a polynomial $P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ is expressed by a row or column vector, like $p = [a_n \quad a_{n-1} \quad \cdots \quad a_1 \quad a_0]$ that contains the coefficients of the polynomial ordered by descending powers. MATLAB has some functions for polynomial manipulations; see Table B3.

TABLE B3 Functions for Polynomials

Function	*Utility*
`conv`	Polynomial multiplication
`deconv`	Polynomial division
`poly`	Convert roots to polynomial
`polyval`	Evaluate a polynomial
`roots`	Find polynomial roots

For the cubic equation

$$r^3 - 2r^2 + 23 = 0$$

its roots are computed by

```
>> roots([1 -2 0 23])
ans =
  2.1550 + 2.3048i
  2.1550 - 2.3048i
 -2.3101
```

The product of polynomials $P_1(x) = 3x^2 + 2x + 5$ and $P_2(x) = 6x^2 + 7x - 1$ is obtained by

```
>> p1=[3 2 5]; p2=[6 7 -1];
>> p=conv(p1,p2)
p =
   18   33   41   33   -5
```

B.3 Graphics

A collection of MATLAB functions is available for sophisticated graphics and visualization of data. Listed in Table B4 are some of them. Use the online help to learn how to use these functions.

TABLE B4 Functions for Graphics

Function	*Utility*
`plot`	Linear plot
`axis`	Axis scaling
`grid`	Add grid lines
`title`	Place graph title
`xlabel`	x-axis label
`ylabel`	y-axis label
`legend`	Graph legend
`subplot`	Divide a graphic window into multiple ones
`plot3`	Plot lines and points in three dimensions

As a demonstrative example, consider function $x = 1 - e^{-2t}(\cos 3t - 0.2 \sin 3t)$. The statements

```
>> t=linspace(0, 4, 101);
>> x = 1-exp(-2*t).*(cos(3*t)-0.2*sin(3*t));
>> plot(t,x)
>> grid; xlabel('t'); ylabel('x')
>> title('Function x vs time t')
```

create 101 data points over the range $0 \leq t \leq 4$ and plot the curve x versus t in Fig. B1.

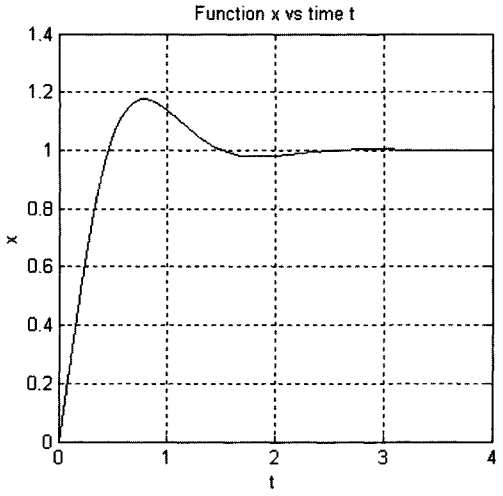


FIGURE B1

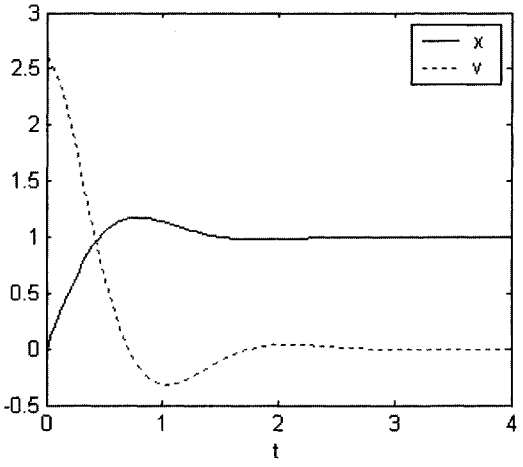


FIGURE B2

Now consider another function $v = e^{-2t}(2.6\cos 3t + 1.6\sin 3t)$. Functions x and v can be plotted in one figure:

```
>> v = exp(-2*t).*(2.6*cos(3*t)+1.6*sin(3*t));
>> plot(t,x,t,v,':')
>> xlabel('t')
>> legend('x', 'v')
```

which produces Fig. B2. In the plot command, symbol `':'` is used to set the dotted line style for the y curve. Some available line styles are given below

Symbol	Line Style
–	solid
:	dotted
–.	dashdot
– –	dashed

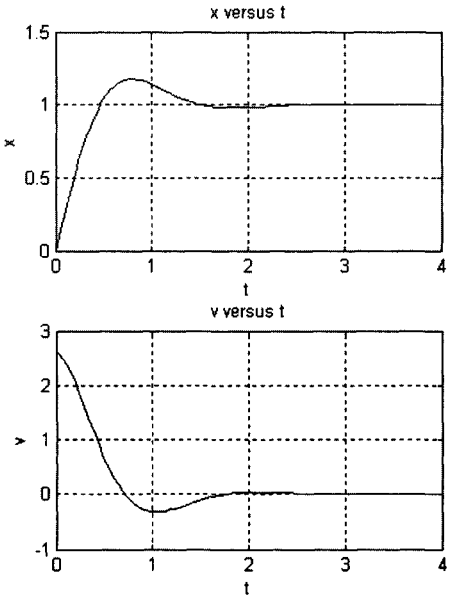


FIGURE B3

Typing `help plot` in the command window shows more information about line styles.

The x and v can also be plotted in two subplots in one figure window:

```
>> subplot(2,1,1)
>> plot(t,x)
>> grid,xlabel('t'), ylabel('x')
>> title('x versus t')
>> subplot(2,1,2)
>> plot(t,v)
>> grid, xlabel('t'), ylabel('v')
>> title('v versus t')
```

which produces the x- and v- curves in two subplots in Fig. B3. Here, function `subplot` partitions the graphic window into two parts, and function `plot` generates a curve for each part.

A three-dimensional line can be generated by the command `plot3(x, y, z)`, where `x`, `y`, `z` are vectors of the same length. For example,

```
>> z=linspace(0,1,201);
>> x=exp(-2*z).*cos(20*z);
>> y = exp(-2*z).*sin(20*z);
>> clf
>> plot3(x,y,z)
>> xlabel('x'), ylabel('y'), zlabel('z')
>> grid
```

produces the curve

$$x = e^{-2t}\cos(20t), \quad y = e^{-2t}\sin(20t), \quad z = t$$

for $0 \leq t \leq 1$; see Fig. B4.

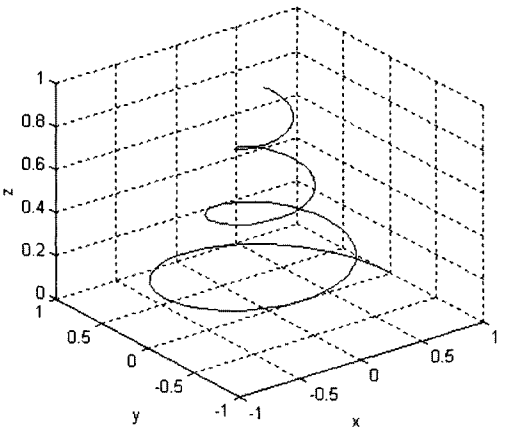


FIGURE B4

The aspect ratio of a plot displayed in the MATLAB figure can be controlled as follows:

`axis equal`	sets the aspect ratio so that the x, y, and z coordinates are equal in scale.
`axis square`	sets the current axis box square in size.
`axis normal`	sets the aspect ratio back to the default setup.

Type `help axis` in the command window to find out more in detail.

B.4 M-Files

In MATLAB, there are two types of M-files: script M-files and function M-files. These files are called *M-files* because their names must end with the extension "`.m`," such as `test.m`.

Script M-Files

MATLAB allows the user to put a number of commands in a simple text file and execute the commands contained in the file as if they were typed at the prompt in the command window. All variables assigned in the file reside in the MATLAB workspace. This type of file is called a *script file*. (Execution of the commands in a script file simply needs to type the name of the M-file in the MATLAB command window). Script M-files are useful for evaluation of a large number of commands in different cases of variable values.

A script file can be created by choosing **New** from the **File** menu and by selecting **M-file**, which launches a text editor. Use this text editor to type the desired MATLAB commands. Choose **Save** or **Save As ...** from the **File** menu to name and save the script file on the computer. The following is an example of script M-file, which computes the free vibration of a spring-mass system for given initial displacement $x0$ and initial velocity $v0$.

```
% freevib.m - a script file for free vibration
% of a spring-mass system
% x0 - initial displacement
% v0 - initial velocity
```

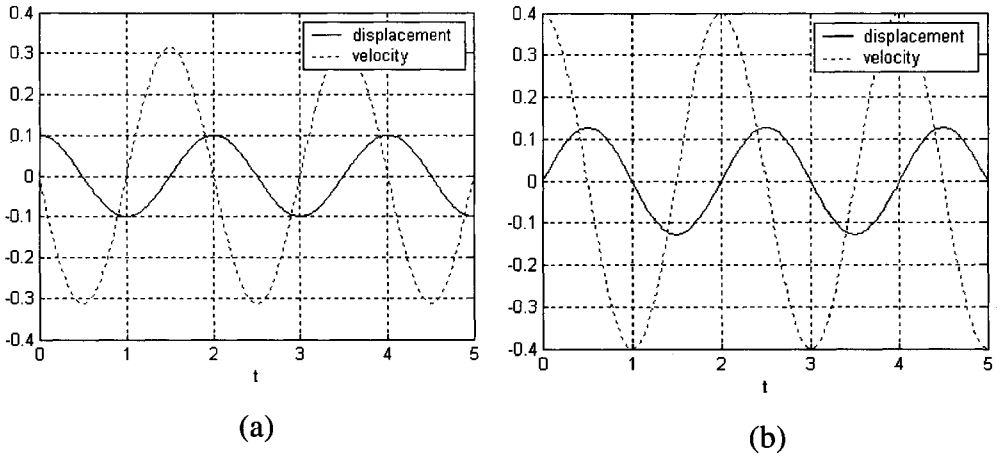


FIGURE B5

```
t=linspace(0,5,201);
wn = pi;      % wn = natural frequency
x = x0*cos(wn*t)+v0/wn*sin(wn*t);
v = -x0*wn*sin(wn*t)+v0*cos(wn*t);
plot(t,x, t,v,':')
grid, xlabel('t')
legend('displacement', 'velocity')
```

In MATLAB, all the text after the percentage sign (%) is taken as a comment statement, and will not be executed.

```
>> x0 = 0.1; v0 = 0;
>> freevib
```

plots the vibration of the system in Fig. B5(a) for x0 = 0.1 and v0 = 0. Also,

```
>> x0 = 0; v0 = 0.4;
>> freevib
```

plots the vibration of the system in Fig. B5(b) for x0 = 0 and v0 = 0.4. In both cases, the script file `freevib.m` has been called to execute the commands in it.

Function M-Files

Function M-files in MATLAB have similar functionalities of subroutines or procedures in other computer programming languages such as Fortran and C. See the following example.

```
function x = frvibration(x0, v0)
% FRVIBRATION - compute and output displacement x of a spring-mass
% system for given initial displacement x0 and initial
% velocity v0
```

```
t=linspace(0,1,10);
wn = pi;      % wn = natural frequency
x = x0*cos(wn*t)+v0/wn*sin(wn*t);
```

Function **frvibration**, stored in a file under the name **frvibration.m**, takes x0 and v0 as input variables and returns vector x as an output variable. For instance,

```
>> frvibration(0.1, 0)
ans =
 0.1000 0.0940 0.0766 0.0500 0.0174 -0.0174 -0.0500 -0.0766 -0.0940 -0.1000
```

A function M-file contains the word **function** at the beginning of the first line, such as

```
function outputvar = FunName(inputvar1, inputvar2, ...)
```

or

```
function [outputvar1, outputvar2, ...] = FunName(inputvar1, inputvar2, ...)
```

where FunName is the function name, `inputvar1, inputvar2, ...` are input variables, and `outputvar, outputvar1, outputvar2, ...` are output variables. The rest of the lines in the function file are commands or comments (texts after the percentage sign %). As a rule, the output variables of the function must be assigned within the function. Also, function FunName should be stored in a file named FunName.m. A function M-file can be created the same way as a script M-file.

Like script M-files, a function M-file must have an extension ".m". However, a function M-file is different from a script file in that the function interacts with the MATLAB workspace only through its input and output variables. The intermediate variables contained in the function do not appear in, or interact with, the MATLAB workspace; they reside locally in a separate workspace of the function. For instance, the intermediate variables of function `frvibration` are `t` and `wn`, which are not created in the MATLAB workspace.

Set MATLAB Search Path

If M-files and data files (files with extension `.mat`) are stored in a folder that is not on the **MATLAB Search Path**, MATLAB cannot find them. In this case, one can add the desired folder to the MATLAB Search Path by choosing the **Set Path ...** from the **File** menu and selecting the **Add Folder ...** button. Another way to resolve this problem is to put those files to be used in the MATLAB **work** folder, which is located inside the MATLAB program folder. To find more about the MATLAB Search Path, go to the **MATLAB Help** from the **Help** menu.

B.5 Control Flow

Like other computer programming languages, MATLAB has some decision-making structures for control of command execution. These decision-making or control-flow structures include **for** loops, **while** loops, and **if-else-end** constructions. Control-flow structures are often used in scrip M-files and function M-files.

for Loops

A `for` loop in general has the form

```
for x = array
    commands
end
```

which allows a set of commands to be repeated a fixed number of times. Here, `array` is a vector or matrix, and the commands between the `for` and `end` statements are executed once for each column of array. For instance,

```
>> for i=1:5
       a(i)=i^2+1;
   end
```

assigns vector a with elements $i^2 + 1$ for $i = 1, 2, \ldots, 5$:

```
>> a
a =
   2   5  10  17  26
```

`for` loops can be nested, as shown in the following example:

```
>> for i = 1:3
      for j = 1:3
          b(i,j) = sin(i*pi/4)*sin(j*pi/4);
      end
   end
>> b
b =
 0.5000  0.7071  0.5000
 0.7071  1.0000  0.7071
 0.5000  0.7071  0.5000
```

which assigns matrix b with elements $b_{ij} = \sin\frac{i\pi}{4}\sin\frac{j\pi}{4}$ for $i, j = 1, 2, 3$.

TABLE B5 Relational and Logical Operators

Relational Operator		*Logical Operator*	
>	less than	&	AND operator
<=	less than or equal to	|	OR operator
>	greater than	~	NOT operator
>=	greater than or equal to		
==	equal to		
~=	not equal to		

Logical Statements

MATLAB has certain rational operators and logical operators for answers to True-False questions in decision making; see Table B5. A logical statement is one that contains relational and

logical operators. The output of a logical statement is **one for True** and **zero for False**. For example,

```
>> y = (5>6)
y =
   0
>> (sin([0 1 2 3 4 5]) > 0) & (cos([0 1 2 3 4 5]) < 0)
ans =
  0   0   1   1   0   0
```

Logical statements are often used in `while` loops and `if-else-end` constructions.

`while` Loops

A `while` loop is used to evaluate a set of commands an indefinite number of times, which is reinforced by a logical statement. A `while` loop has the form

```
while expression
   commands
end
```

where `expression` is a logical statement. The commands between the `while` and `end` statements are executed as long as the logical statement is true. For example,

```
>> n = 1; y = 2;
>> while y+0.00001 < exp(1)
      n=n+1;
      y = (1+1/n)^n;
   end
>> format long
>> y, n
y =
  2.71827182846131
n =
   135913
```

The above `while` loop computes the expression $y = \left(1 + \frac{1}{n}\right)^n$ for a large enough n such that $y + 0.00001 \geq e$, where e is the irrational number $2.718281828459\cdots$.

`if-else-end` Constructions

An `if-else-end` construction is used for conditional evaluation of several sequences of commands. The general form of an `if-else-end` construction is

```
if condition1
    commands evaluated if condition1 is True
elseif condition2
    commands evaluated if condition2 is True
elseif ...
    ⋮
else
    commands evaluated if none of the about expressions are True
end
```

where `condition1, condition2 ...` are logical statements. Two simplified versions of the above `if-else-end` construction are

```
if condition
       commands evaluated if condition is True
end
```

and

```
if condition
       commands evaluated if condition is True
else
       commands evaluated if condition is False
end
```

The above-mentioned forms of `if-else-end` constructions can be combined and nested in use.

As an example, consider the following function M-file:

```
function x = quadeq(a, b, c)
d=b^2-4*a*c;
disp(' ')
if a == 0
    disp('This is not a quadratic equation'); return
end
if d > 0
    disp('The equation has two different real roots')
elseif d == 0
    disp('The equation has two identical real roots')
else
   if b == 0
       disp('The equation has two imaginary roots')
   else
       disp('The equation has two complex roots')
   end
end
x1 = (-b+sqrt(d))/2/a; x2 = (-b-sqrt(d))/2/a;
x = [x1; x2];
```

which uses `if-else-end` constructions to tell the type of the roots for a quadratic equation $ax^2 + bx + c = 0$.

```
>> x=quadeq(1,2,3)
```

The equation has two complex roots

```
x =
```

```
 -1.0000 + 1.4142i
 -1.0000 - 1.4142i
>> x=quadeq(1,4,4)
The equation has two identical real roots
x =
   -2
   -2
```

B.6 Solution of Algebraic and Differential Equations

MATLAB provides several useful functions for the solution of algebraic and differential equations; see Table B6, where `inv`, `roots`, `fzero`, `fmins`, `ode23`, and `ode45` are MATLAB functions, and \ is the left division operator. Use the online help to find how to use these functions.

TABLE B6 Solution of Algebraic and Differential Equations

Equation Type	*MATLAB Command*

System of linear algebraic equations
$Ax = y$
	`x = inv(A)*x`
	or
	`x = A\y`

Polynomial equation $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = 0$	`x = roots([`$a_n \quad a_{n-1} \quad \cdots \quad a_1 \quad a_1$`])`
Nonlinear algebraic equation of one variable $f(x) = 0$	`x = fzero('FUN', x0)`
Nonlinear algebraic equation of multiple variables $f(x) = 0$ where x is a vector.	`x = fmins('FUN', x0)`

Linear and nonlinear differential equations
$\frac{d}{dt}x = f(x,t)$
where x is either a scalar or a vector.
	`[t,x] = ode23('FUN', tspan, x0)`
	or
	`[t,x] = ode45('FUN', tspan, x0)`

EXAMPLE 1.

For the system of linear algebraic equations

$$\begin{bmatrix} 1 & -5 & 2 \\ 0 & 3 & -4 \\ 4 & 7 & 10 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \\ 8 \end{pmatrix}$$

the solution is obtained by

```
>> A=[1 -5 2; 0 3 -4; 4 7 10];
>> f=[3; -1; 8];
>> x=A\f
x =
   1.9474
  -0.1579
   0.1316
```

The solution can also be obtained by `inv(A)*f`.

EXAMPLE 2.

To solve the nonlinear algebraic equation

$$f(x) = 2e^{-x} - \sin x = 0$$

for the smallest root, the following function M-file is generated for function `fzero`:

```
function f = NLfunc(x)
f = 2*exp(-x)-sin(x);
```

The initial guess of the solution is $x_0 = 0$. A root of the nonlinear equation is obtained by

```
>> fzero('NLfunc', 0)
ans =
  0.9210
```

The commands

```
>> x=linspace(0,2,2001);
>> f = NLfunc(x);
>> plot(x,f)
>> grid; xlabel('x'); ylabel('f')
```

plot the function $f(x)$ versus x in Fig. B6, which shows that $x = 0.9210$ is the smallest root.

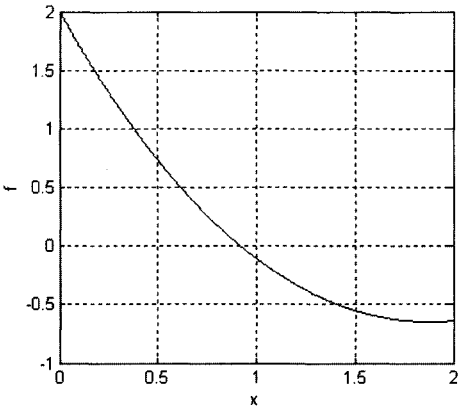


FIGURE B6

EXAMPLE 3.

The differential equation

$$\ddot{x} + \dot{x} + 2\sin(x) = 0$$
$$x(0) = 0.2, \quad \dot{x}(0) = 0.5$$

is solved by the MATLAB function ode45. To this end, the equation is rewritten as the first-order state equations

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_2 - 2\sin(x_1)$$

with the initial conditions $x_1(0) = 0.2$, $x_2(0) = 0.5$, where $x_1 = x$ and $x_2 = \dot{x}$. The following function M-file, which is required by function ode45, is created based on the state equations:

```
function f = sfunc(t,x)
f = zeros(2,1);
f(1) = x(2);
f(2) = -x(2)-2*sin(x(1));
```

The commands

```
>> [t,x]=ode45('sfunc', [0 5], [0.2 0.5]);
>> plot(t,x(:,1),t,x(:,2),':')
>> grid; xlabel('t'),
>> legend('x1','x2')
```

yield x_1 and x_2 for $0 \leq t \leq 5$, which are plotted against t in Fig. B7. Note that the execution of the `ode45` command calls for the function `sfunc`.

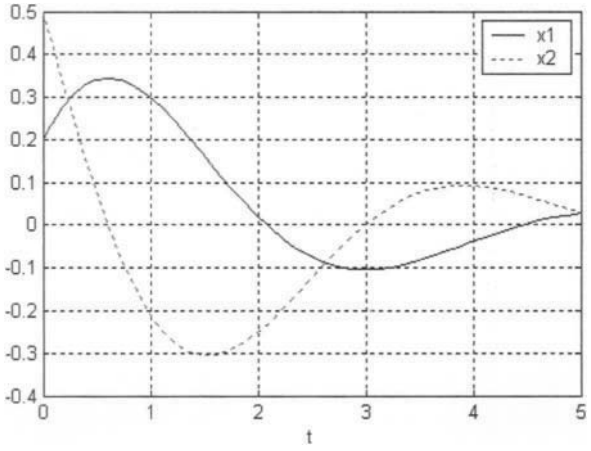


FIGURE B7

B.7 Control System Toolbox

Besides the general features and functions described previously, MATLAB provides collections of function M-files called *toolboxes*, which solve problems in specific areas or disciplines. Listed in Table B7 are some functions from the *Control System Toolbox*. These functions are useful in design of feedback controllers for mechanical systems and flexible structures; see Chapters 11 and 12, for instance.

TABLE B7 Functions from the Control System Toolbox

Transfer Function Formulation	
`tf`	Create a transfer function model
`zpk`	Create a zero-pole-gain model
`series`	Series interconnection of two blocks (transfer functions)
`parallel`	Parallel interconnection of two blocks (transfer functions)
`feedback`	Obtain the transfer function of a feedback control system
State-Space Formulation	
`ss`	Create a state-space model
`ss2tf`	State-space to transfer function conversion
`tf2ss`	Transfer function to state-space conversion
`minreal`	Minimal realization and pole-zero cancellation
`ctrb`	Compute controllability matrix
`obsv`	Compute observability matrix

TABLE B7 Functions from the Control System Toolbox *(continued)*

System Response	
`impulse`	Impulse response
`step`	Step response
`lsim`	Response to arbitrary inputs
Root Locus	
`rlocus`	Root locus
`sgrid`	Generate an s plane grid of constant damping factors and natural frequencies
Frequency Response	
`freqresp`	Frequency response
`bode`	Bode plot of the frequency response
`nyquist`	Nyquist plot
`margin`	Gain and phase margins
`nichols`	Nichols frequency response
Controller Design Tools	
`rlocfind`	Interactive root locus gain determination
`rltool`	SISO system design tool via root locus
`sisotool`	Open the SISO Design Tool
`place`	Pole placement technique
`lqr`	Linear-quadratic regulator design

EXAMPLE B.1

Consider a dynamic system described by the transfer function

$$G(s) = \frac{Y(s)}{R(s)} = \frac{s + 2.5}{2s^3 + 5s^2 + 20s + 18}$$

The response of the system under a unit impulse input $r(t) = \delta(t)$ is plotted in Fig. B8 by

```
>> num = [1 2.5];
>> den = [2 5 20 18];
>> impulse(num,den)
```

Here vectors num and den contain the coefficients of the numerator and denominator of the transfer function, respectively.
Furthermore, let the system be under feedback control. Assume that the closed-loop characteristic equation is

$$1 + k \cdot G(s) = 0$$

where k is a positive feedback gain parameter. The closed-loop root loci versus the parameter k are plotted in Fig. B9 by

```
>> sys = tf(num, den)
>> rlocus(sys)
```

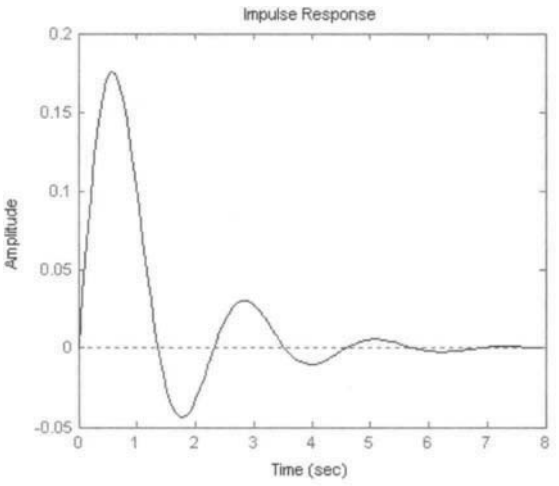


FIGURE B8

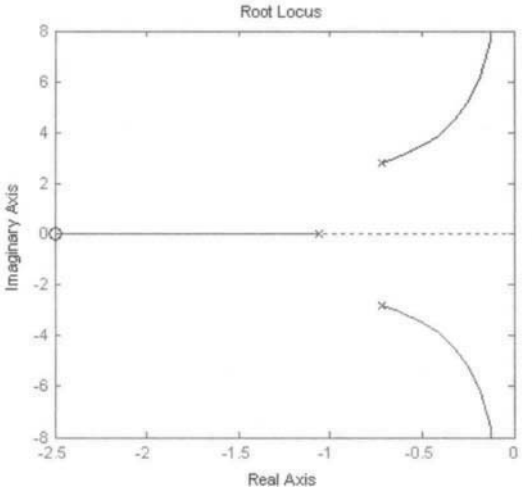


FIGURE B9

Appendix C The Distributed Transfer Function Method

Inside
- DTFM for One-Dimensional Continua
- Transfer Function Synthesis of Multibody Structures
- DTFM for Two- and Three-Dimensional Problems
- References

Many MATLAB functions of this book are developed according to the Distributed Transfer Function Method (DTFM), which is a closed-form analytical solution method for modeling and analysis of solids and structures. In this appendix, the basic concepts of the DTFM are briefly presented. The reader may refer to Section C.4 for more information on the method.

C.1 DTFM for One-Dimensional Continua

Governing Equations

Consider a uniform one-dimensional continuum of length L in Fig. C1, where $w(x,t)$ is the generalized displacement of the continuum and $f(x,t)$ is a generalized force. The continuum is a model of various flexible systems such as beams, bars, beam-columns, strings, flexible rotating shafts, and axially moving belts. The response of the continuum is governed by the partial differential equation

$$\sum_{k=0}^{n}\left(a_k\frac{\partial^2}{\partial t^2}+b_k\frac{\partial}{\partial t}+c_k\right)\frac{\partial^k}{\partial x^k}w(x,t)=f(x,t),\quad 0\le x\le L \tag{C.1}$$

that is subject to the boundary conditions

$$M_j w(x,t)|_{x=0}+N_j w(x,t)|_{x=L}=\gamma_{Bj}(t),\quad j=1,2,\ldots,n \tag{C.2}$$

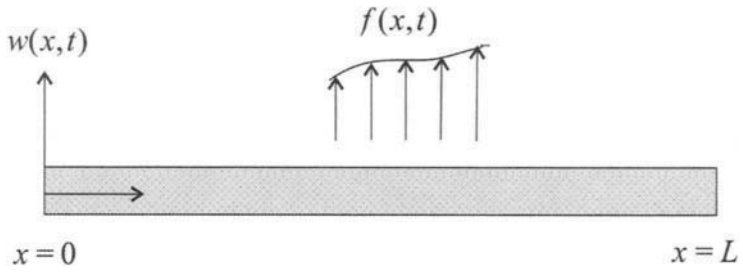


FIGURE C1 One-dimensional continuum.

and the initial conditions

$$w(x,t)|_{t=0} = u_0(x), \quad \frac{\partial}{\partial t} w(x,t)|_{t=0} = v_0(x) \tag{C.3}$$

Here a_k, b_k, and c_k describe the spatial distributions of system parameters, such as inertia, damping, gyroscopic forces, stiffness, and axial loading; M_j and N_j are differential operators; $\gamma_{Bj}(t)$ are the boundary disturbances; and $u_0(x)$ and $v_0(x)$ are the initial displacement and velocity of the continuum.

Spatial State Formulation

Laplace transform of Eq. (C.1) with respect to time t leads to

$$\frac{\partial^n}{\partial x^n}\bar{w}(x,s) = \sum_{k=0}^{n-1} d_k(s)\frac{\partial^k}{\partial x^k}\bar{w}(x,s) + \bar{f}(x,s) + \bar{f}_I(x,s), \quad 0 \le x \le L \tag{C.4}$$

where the over-bar stands for Laplace transformation, s is the complex Laplace transform parameter,

$$d_k(s) = -\frac{a_k s^2 + b_k s + c_k}{a_n s^2 + b_n s + c_n}, \quad k = 0, 1, \ldots, n-1$$

and $\bar{f}_I(x,s)$ represents the terms related to the initial disturbances $u_0(x)$ and $v_0(x)$. Also, Laplace transform of the boundary conditions (C.2) gives

$$\bar{M}_j\bar{w}(x,s)|_{x=0} + \bar{N}_j w(x,s)|_{x=L} = \bar{\gamma}_{Bj}(s) + \bar{\gamma}_{Ij}(s) \equiv \gamma_j(s), \quad j = 1, 2, \ldots, n \tag{C.5}$$

where $\bar{M}_j$ and $\bar{N}_j$ are operators M_j and N_j with $\partial/\partial t$ contained in them replaced by s, and $\bar{\gamma}_{Ij}(s)$ represents the initial disturbances at the boundaries.

By defining the spatial state vector

$$\eta(x,s) = \left(\bar{w}(x,s) \quad \frac{\partial}{\partial x}\bar{w}(x,s) \quad \cdots \quad \frac{\partial^{n-1}}{\partial x^{n-1}}\bar{w}(x,s)\right)^T \tag{C.6}$$

Eqs. (C.4) and (C.5) are cast into the following matrix form

$$\frac{\partial}{\partial x}\eta(x,s) = F(s)\eta(x,s) + q(x,s), \quad 0 \le x \le L \tag{C.7}$$

subject to the boundary condition

$$M_b(s)\eta(0,s) + N_b(s)\eta(L,s) = \gamma(s) \tag{C.8}$$

where

$$F(s) = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ d_0(s) & d_1(s) & \cdots & \cdots & d_{n-1}(s) \end{bmatrix}, \quad q(x,s) = \frac{\bar{f}(x,s) + \bar{f}_I(x,s)}{a_n s^2 + b_n s + c_n} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad \gamma(s) = \begin{pmatrix} \gamma_1(s) \\ \gamma_2(s) \\ \vdots \\ \gamma_n(s) \end{pmatrix}$$

and $M_b(s)$ and $N_b(s)$ are $n \times n$ matrices composed of the coefficients of the operators $\bar{M}_j$ and $\bar{N}_j$.

As an example, consider an axially moving string whose transverse displacement $z(x,t)$ is governed by the differential equation

$$\rho\left\{\frac{\partial^2}{\partial t^2} + 2c\frac{\partial^2}{\partial x \partial t} + c^2\frac{\partial^2}{\partial x^2}\right\} z(x,t) - T\frac{\partial^2}{\partial x^2} z(x,t) = f(x,t), \quad 0 \le x \le L \tag{C.9}$$

where ρ, c, and T are the linear density (mass per unit length), translational speed, and tension of the string, respectively. The above equation is converted into the s-domain state form (C.7) with

$$\eta(x,s) = \begin{pmatrix} \bar{z}(x,s) \\ \dfrac{\partial}{\partial x}\bar{z}(x,s) \end{pmatrix}, \quad F(s) = \begin{bmatrix} 0 & 1 \\ \dfrac{\rho s^2}{T - \rho c^2} & \dfrac{2\rho c s}{T - \rho c^2} \end{bmatrix}, \quad q(x,s) = \frac{\bar{f}(x,s) + \bar{f}_I(x,s)}{\rho c^2 - T} \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

If the string has the fixed-free boundary conditions

$$z(0,t) = 0, \quad \frac{\partial}{\partial x} z(L,t) = 0$$

the boundary matrices in Eq. (C.8) are

$$M_b(s) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad N_b(s) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Eigenvalue Problem

With vanishing external and boundary disturbances, Eqs. (C.7) and (C.8) become

$$\begin{aligned} \frac{\partial}{\partial x}\eta(x,s) &= F(s)\eta(x,s), \quad 0 \le x \le L \\ M_b(s)\eta(0,s) + N_b(s)\eta(L,s) &= 0 \end{aligned} \tag{C.10}$$

which define an eigenvalue problem for the continuum. The solution of Eq. (C.10) is of the form

$$\eta(x,s) = e^{F(s)x}\,\eta(0,s), \quad 0 \le x \le L \tag{C.11}$$

where $e^{F(s)x}$ is the exponential of matrix $F(s)x$, and vector $\eta(0,s)$ is the solution of the homogeneous equation

$$\left[M_b(s) + N_b(s)e^{F(s)L}\right]\eta(0,s) = 0. \tag{C.12}$$

(Refer to Appendix A.9 for the definition and properties of exponential matrices.) The eigenvalues of the continuum are the roots of the characteristic equation

$$\det\left[M_b(s) + N_b(s)e^{F(s)L}\right] = 0 \tag{C.13}$$

which in general is of transcendental form.

If the eigenvalue problem is derived from the free vibration of an undamped flexible system, as in Chapters 12 and 13, the roots of Eq. (C.13) are of the form

$$s = j\omega_k, \quad j = \sqrt{-1}, \quad k = 1, 2, \ldots$$

where ω_k are the kth natural frequency of the continuum. The mode shape corresponding to ω_k can be obtained by solving Eq. (C.12) for a nonzero $\eta(0,s)$ at $s = j\omega_k$ and substituting the result into Eq. (C.11).

If the eigenvalue problem comes from the buckling of a structure, as presented in Chapter 4, the Laplace transform parameter s = 0, and matrices F, M_b, and N_b are functions of a buckling load parameter p. In this case, the characteristic equation is

$$\Delta(p) \equiv \det\left[M_b(0) + N_b(0)e^{F(0)L}\right] = 0 \tag{C.14}$$

whose roots are the buckling loads of the continuum. The associate mode shapes can be similarly obtained from Eqs. (C.11) and (C.12).

Response by Distributed Transfer Functions

The solution of Eqs. (C.7) and (C.8), by References 1 and 2, is given by

$$\eta(x,s) = \int_0^L G(x,\xi,s)q(\xi,s)\,d\xi + H(x,s)\gamma(s) \tag{C.15}$$

where $G(x,\xi,s)$ and $H(x,s)$ are called the *distributed transfer functions* of the continuum, and are given by

$$G(x,\xi,s) = \begin{cases} H(x,s)M_b(s)e^{-F(s)\xi} & \text{for } x \geq \xi \\ -H(x,s)N_b(s)e^{F(s)(L-\xi)} & \text{for } x \leq \xi \end{cases}$$
$$H(x,s) = e^{F(s)x}\left(M_b(s) + N_b(s)e^{F(s)L}\right)^{-1} \tag{C.16}$$

In the above response solution, no approximation or series truncation is made.

The above distributed transfer function formulation provides a new way to determine static and dynamic response of elastic continua in exact and closed form. To determine the static response of a continua, as seen in Chapters 2 to 4, 6, and 8, simply set $s = 0$ in Eq. (C.15). To determine the steady-state response of a continuum subject to a harmonic excitation of frequency Ω, as seen in Chapters 12 and 13, replace s by $j\Omega$ in Eq. (C.15). To evaluate the transient (time-domain) response of a continuum, inverse Laplace transform of Eq. (C.15) is required. Reference 10 presents a closed-form transient response solution based on the inverse Laplace transform of the distributed transfer functions.

Nonuniform Continua

The above-mentioned DTFM is extended to continua whose coefficients, like a_k, b_k, and c_k in Eq. (C.1), are functions of spatial coordinate x. For a nonuniform continuum, the transfer function formulation (C.15) is still valid, if $e^{F(s)x}$ in Eq. (C.16) is replaced by a fundamental matrix $\Phi(x, s)$; see Reference (2). Reference (4) presents two techniques for evaluating $\Phi(x, s)$. The MATLAB functions for buckling analysis of nonuniform columns in Chapter 4 are developed based on one of the techniques.

C.2 Transfer Function Synthesis of Multibody Structures

The DTFM is capable of obtaining closed-form analytical solutions for flexible structures of multiple components. This system-level modeling and analysis is referred as to *transfer function synthesis* (see Reference 3). The transfer function synthesis takes the following three steps:

Step 1. Transfer Function Representation

The structure in consideration is decomposed into a number of one-dimensional components. The points where the components are interconnected are called *nodes*. The response of the components is expressed by their distributed transfer functions. For instance, for a component with nodes i and j as its two ends, its response by Eq. (C.15) is

$$\eta(x, s) = \eta_f(x, s) + H(x, s)\gamma(s) \tag{C.17}$$

where

$$\eta_f(x, s) = \int_0^L G(x, \xi, s) q(\xi, s)\, d\xi, \quad \gamma(s) = \begin{pmatrix} \alpha_i(s) \\ \alpha_i(s) \end{pmatrix}$$

and $\alpha_i(s)$ and $\alpha_j(s)$ are vectors of displacements at nodes i and j, respectively.

Step 2. Assembly of Components

Let the displacements of the structure at node i in the global coordinates be represented by vector $u_i(s)$, where the parameter s indicates that the displacements have been Laplace transformed with respect to time. The $u_i(s)$ is related to $\alpha_i(s)$ in Eq. (C.17) by a coordinate transformation. Suppose that at node i components (A), (B),..., (D) are interconnected, as illustrated in Fig. C2. Force balance at the node requires

$$Q_A(s) + Q_B(s) + \cdots + Q_D(s) - C_i(s)u_i(s) + p_i(s) = 0 \tag{C.18}$$

where $Q_A(s)$, $Q_B(s)$, $\cdots$, $Q_D(s)$ are vectors of the forces applied to the node by components (A), (B),..., (D), respectively; the term $C_i(s)u_i(s)$ describes the forces of any constraints such as springs, dampers, and lumped masses that are imposed at the node; and $p_i(s)$ is the vector of external forces applied at Node i. By Eq. (C.17), vectors $Q_A(s)$, $Q_B(s)$, $\cdots$, $Q_D(s)$ can be expressed in terms of $u_i(s)$, $u_j(s)$, $\cdots$, $u_l(s)$. Accordingly, Eq. (C.18) is reduced to

$$K_{ii}(s)u_i(s) + K_{ij}(s)u_j(s) + \cdots + K_{il}(s)u_l(s) = q_i(s) \equiv p_i(s) + f_i^{TR}(s) \tag{C.19}$$

where $K_{ii}(s), K_{ij}(s), \cdots, K_{il}(s)$ are matrices of dynamic stiffness that are composed of the elements of the distributed transfer functions given in Eq. (C.16), and $f_i^{TR}(s)$ is the vector

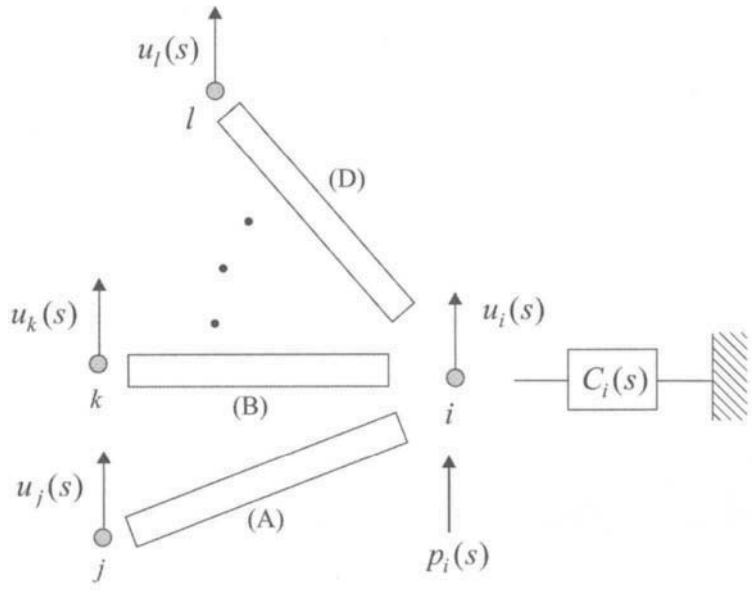


FIGURE C2 Interconnection of components (A), (B), ..., (D) at Node *i*.

of transmitted forces due to external forces applied at the interior points of the components that are connected to node *i*. Force balance at all nodes leads to a global dynamic equilibrium equation

$$K(s)u(s) = q(s) \tag{C.20}$$

where matrix $K(s)$ is the global dynamic stiffness matrix; $u(s)$ is the global nodal displacement vector, and $q(s)$ is the global nodal force vector.

Step 3. Static and Dynamic Analysis

With the equilibrium equation (C.20), exact and closed-form solutions can be obtained for various static and dynamic problems of structures.

Static Analysis By setting $s = 0$, the nodal displacements are obtained as

$$u(0) = K^{-1}(0)q(0) \tag{C.21}$$

Substituting (C.21) into Eq. (C.15) yields the static response of every component.

Eigensolutions For free vibration of a structure, Eq. (C.20) becomes

$$K(s)u(s) = 0 \tag{C.22}$$

which defines an eigenvalue problem. The characteristic equation of the structure is

$$\det K(s) = 0 \tag{C.23}$$

the roots of which are the eigenvalues of the structure. The mode shape associated with an eigenvalue is determined by solving Eq. (C.22) for a nonzero vector $u(s)$. In buckling analysis of structures, buckling loads and associated mode shapes can be determined in a similar manner.

Frequency Response For a structure under harmonic excitations of frequency Ω, the nodal displacements are determined as

$$u(j\Omega) = K^{-1}(j\Omega)q(j\Omega), \quad j = \sqrt{-1} \tag{C.24}$$

Substituting (C.24) into Eq. (C.15) gives the frequency response or steady-state response of the structure at any point.

The above-mentioned transfer function synthesis is further detailed in Reference 3, and has been applied to buckling analysis of constrained stepped columns Reference 14 and dynamic analysis of complex flexible rotor systems Reference 11. Also, in Chapter 8 of this book, the transfer function synthesis is applied to static analysis of frames.

C.3 DTFM for Two- and Three-Dimensional Problems

The DTFM can be extended to two- and three-dimensional continua. As an illustrative example, consider a rectangular elastic region in Fig. C3. Divide the region into N strips by N+1 lines, which are called *nodal lines*. Along the jth nodal line, define a nodal line displacement vector $\phi_j(x,t)$. Interpolate the displacements of the jth strip along its lateral (y-) direction by

$$\{U(x,y,t)\} = [N(y)] \begin{pmatrix} \phi_j(x,t) \\ \phi_{j+1}(x,t) \end{pmatrix} \tag{C.25}$$

where $[N(y)]$ is a matrix of polynomial functions. The displacement interpolation, by Hamilton's principal or other means, reduces the governing equations of the elastic region to the matrix partial differential equation

$$\left([M]\frac{\partial^2}{\partial t^2} + [K_2]\frac{\partial^2}{\partial x^2} + [K_1]\frac{\partial}{\partial x} + [K_0]\right)\{\phi(x,t)\} = \{f(x,t)\}, \quad 0 \le x \le a \tag{C.26}$$

where $[M]$ and $[K_i]$ are matrices related to the inertia and stiffness of the elastic region, $\{\phi\}$ is the vector of independent nodal line displacements, and $\{f\}$ is the vector of equivalent

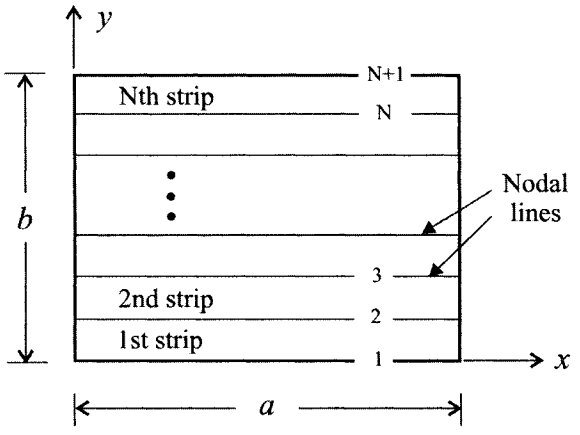


FIGURE C3 A rectangular elastic region divided into strips.

external forces. Without loss of generality, assume zero initial conditions. Laplace transform of Eq. (C.26) and use of spatial formulation in Section C.1 yields Eq. (C.7), with

$$\eta(x,s) = \begin{pmatrix} \{\bar{\phi}(x,s)\} \\ \frac{\partial}{\partial x}\{\bar{\phi}(x,s)\} \end{pmatrix}, \quad q(x,s) = \begin{pmatrix} 0 \\ [K_2]^{-1}\{\bar{f}(x,s)\} \end{pmatrix}$$
$$F(s) = \begin{bmatrix} 0 & [I] \\ -[K_2]^{-1}(s^2[M] + [K_0]) & -[K_2]^{-1}[K_1] \end{bmatrix} \tag{C.27}$$

where $[I]$ is an identity matrix. The solution of the elastic region is then given by Eq. (C.15). Furthermore, the transfer function synthesis presented in Section C.2 can be generalized for an elastic region composed of multiple rectangular subregions. Reference 8 shows the results of static and dynamic analyses of a square region and an L-shaped region.

The above strip DTFM can be applied to rectangular plates; see Reference 6. The DTFM is also valid for continua of other geometries, such as circular and sectorial plates (Reference 9), ring-stiffened cylindrical shells (Reference 5), thick laminated composite shells and panels (Reference 7), plates with curved boundaries (Reference 12), and prismatic elastic multibody solids (Reference 13). In these analyses, distributed transfer functions deliver highly accurate analytical or semi-analytical solutions.

C.4 References

1. Yang B 1992 "Transfer Functions of Constrained/Combined One-Dimensional Continuous Dynamic Systems," *Journal of Sound and Vibration*, Vol. 156: No. 3, pp. 425–443.
2. Yang B, Tan C A 1992 "Transfer Functions of One-Dimensional Distributed Parameter Systems," *ASME Journal of Applied Mechanics*, Vol. 59: No. 4, pp. 1009–1014.
3. Yang B 1994 "Distributed Transfer Function Analysis of Complex Distributed Parameter Systems," *ASME Journal of Applied Mechanics*, Vol. 61: No. 1, pp. 84–92.
4. Yang B, Fang H 1994 "Transfer Function Formulation of Nonuniformly Distributed Parameter Systems," *ASME Journal of Vibration and Acoustics*, Vol. 116: No. 4, pp. 426–432.
5. Yang B, Zhou J 1995 "Analysis of Ring-Stiffened Cylindrical Shells," *ASME Journal of Applied Mechanics*, Vol. 62: No. 4, pp. 1005–1014.
6. Zhou J, Yang B 1996 "Strip Distributed Transfer Function Method for the Analysis of Plates," *International Journal of Numerical Methods in Engineering*, Vol. 39: No. 11, pp. 1915–1932.
7. Zhou J, Yang B 1996 "Three-Dimensional Stress Analysis of Thick Laminated Composite Cylindrical Shells and Panels," *AIAA Journal*, Vol. 34: No. 9, pp. 1960–1964.
8. Yang B, Zhou J 1996 "Semi-analytical Solution of 2-D Elasticity Problems by the Strip Distributed Transfer Function Method," *International Journal of Solid and Structures*, Vol. 33: No. 27, pp. 3983–4005.
9. Yang B, Zhou J 1997 "Strip Distributed Transfer Function Analysis of Circular and Sectorial Plates," *Journal of Sound and Vibration*, Vol. 201: No. 5, pp. 641–647.
10. Yang B, Wu X 1997 "Transient Response of One-Dimensional Distributed Systems: a Closed-Form Eigenfunction Expansion Realization," *Journal of Sound and Vibration*, Vol. 208: No. 5, pp. 763–776.
11. Fang H, Yang B 1998 "Modeling, Synthesis, and Dynamic Analysis of Complex Flexible Rotor Systems," *Journal of Sound and Vibration*, Vol. 211: No. 4, pp. 571–592.

12. Yang B, Park D-H 1999 "Analysis of Plates with Curved Boundaries Using Isoparametric Strip Distributed Transfer Functions," *International Journal of Numerical Methods in Engineering*, Vol. 44: pp. 131–146.
13. Park D-H, Yang B 2001 "Static and Vibration Analyses of Prismatic Elastic Multibody Solids," *International Journal of Structures and Structural Dynamics*, Vol. 14: No. 1, pp. 154–162.
14. Yang B, Park D-H 2003 "Exact Buckling Analysis of Constrained Stepped Columns," *International Journal of Structural Stability and Dynamics*, Vol. No. 2: pp. 143–167.

This Page Intentionally Left Blank

Appendix D Conversion of Units

This appendix presents the prefixes for the standard international system (SI) in Table D1 and conversion between the standard international system and U.S. customary system in Table D2.

TABLE D1 SI Unit Prefixes

Multiplication Factor	*Prefix*	*Symbol*
10^{12}	tera	T
10^{9}	giga	G
10^{6}	mega	M
10^{3}	kilo	k
10^{2}	hecto	h
10	deka	da
10^{-1}	deci	d
10^{-2}	centi	c
10^{-3}	milli	m
10^{-6}	micro	μ
10^{-9}	nano	n
10^{-12}	pico	p

TABLE D2 Conversion of Units

Quantity	SI Unit	US Unit	Conversion
Time	s (second)	sec (second)	
Mass	kg	lb mass	1 kg = 2.2046 lb mass 1 lb mass = 0.4536 kg
		slug	1 kg = 0.0685 slug 1 slug = 32.174 lb mass = 14.594 kg
Length	m	ft	1 m = 3.2808 ft 1 ft = 0.3048 m
	km	mile	1 kg = 1000 m = 0.6214 mile 1 mile = 5280 ft = 1.6093 kg
Area	m^2	ft^2	1 m^2= 10.7639 ft^2 1 ft^2 = 0.0929 m^2
		in^2	1 m^2= 1550.0031 in^2 1 in^2 = 0.00064516 m^2
Volume	m^3	ft^3	1 m^3 = 35.3147 ft^3 1 ft^3 = 0.0283168 m^3
	cm^3	in^3	1 cm^3 = in^3 1 in^3 = cm^3
Volume—liquids	L (liter)	gal	1 L = 0.001 m^3 = 0.26418 gal 1 gal = 3.7853 L
Force	N (Newton)	lb	1 N = 1 kg·m/s^2 = 0.2248 lb 1 lb = 4.4482 N
Moment of force or torque	N·m	lb-ft	1 N·m = 0.7376 lb-ft 1 lb-ft = 1.3558 N·m
Energy or work	J (Joule)	ft-lb	1 J = 1 N·m = 0.7376 ft-lb 1 ft-lb = 1.3558 J
Power	W (Watt)	ft-lb/sec	1 W = 0.7376 ft-lb/sec 1 ft-lb/sec = 1.3558 W
		hp	1 hp = 745.7 W 1 W = 0.00134102 hp
Impulse	N·s	lb-sec	1 N·s = 0.2248 lb-sec 1 lb-sec = 4.4482 N·s
Momentum	kg·m/s	lb-sec	1 kg·m/s = 0.2248 lb-sec 1 lb-sec = 4.4482 kg·m/s
Mass moment of inertia	kg·m^2	ft-lb-sec^2	1 kg·m^2 = 0.7376 ft-lb-sec^2 1 ft-lb-sec^2 = 1.3558 kg· m^2
Pressure or stress or elastic modulus	Pa (Pascal)	lb/ft^2 psi	1 Pa = 1 N/ m^2 = 0.2248 lb/ft^2 1 lb/ ft^2 = 4.4482 Pa 1 psi = 6,894.757 Pa

SI = the standard international system
US = the U.S. customary system

Appendix E Mechanical Properties of Engineering Materials

This appendix summarizes the mechanical properties of selected engineering materials in Tables E1 and E2, in the standard international system (SI) of units and the U.S. customary system (U.S.) of units, respectively. The data provided here are average values or approximations of material coefficients. Exact values may vary greatly, depending on composition, heat treatment, and mechanical working. Therefore, the tables should be used only for educational purposes. For professional practice, obtain material properties from the supplier of materials.

Notation

ρ = density (mass per unit volume)
E = modulus of elasticity or Young's modulus
G = modulus of rigidity or shear modulus
ν = Poisson's ratio
σ_Y = yield stress in tension and compression
σ_u = ultimate stress in tension and compression
α = coefficient of thermal expansion

Units

GPa = 10^9 Pa = 10^9 N/m^2
MPa = 10^6 Pa = 10^6 N/m^2
1 psi = 1 lb/in^2 = 144 lb/ft^2
1 slug /ft^3 = 0.018619 lb mass/in^2
°C = Celsius temperature (°C = 5/9 × [°F − 32])
°F = Fahrenheit temperature

TABLE E1 Material Properties in SI Units

Material	ρ ($10^3 kg/m^3$)	E (GPa)	G (GPa)	ν	σ_Y [1] (MPa)	σ_u [2] (MPa)	α (10^{-6}/°C)
Aluminum	2.7	70	26	0.33	20	70	23.9
Aluminum alloys							
2014-T6	2.70	73.1	27	0.35	414	469	23
6061-T6	2.71	68.9	26	0.35	255	290	24
Cast iron alloys							
Gray ASTM 20	7.19	67	27	0.28	–	179 (669)	12
Malleable ASTM A197	7.28	172	68	0.28	–	276 (572)	12
Concrete							
Low strength	2.38	22.1	–	0.15	[12]	–	11
High strength	2.38	29.0	–	0.15	[38]	–	11
Copper							
Soft (pure)	8.92	110	41	0.34	69	220	17.6
Hard (Be-Cu)	8.4	124	47	0.34	1,170	1,310	17.8
Glass	2.63	69	28	0.25	–	(69–862)	5.4–16.2
Magnesium alloy Am 1004-T61	1.83	44.7	18	0.30	152	276	26
Nickel steel	7.73	200	78	0.29	138–207	310–414	13
Plastic–Kevlar 49	1.45	131	–	0.34	–	717(483)	–
Structural steel							
A36	7.83	200	78	0.28	248	400	12.1
A572	7.83	200	78	0.28	345	448	12.1
Stainless steel							
S40500	7.73	191	73	0.31	172	414	17.3
S17700	7.73	191	73	0.31	1,030	1,280	17.3
Titanium	4.54	110	40	0.33	400	500	8.5
Titanium alloy 5 Al, 2.5 Sn	4.43	117	44	0.34	758	793	10.3
Woods							
Douglas fir	0.50	12.1	–	–	–	(51)	–
Southern pine	0.58	12.1	–	–	–	(58)	–

1. Shear yield stress noted in brackets [ ].
2. Compression ultimate stress noted in parentheses ( ).

TABLE E2 Material Properties in U.S. Units

Material	ρ (slug/ft^3)	E (10^6 psi)	G (10^6 psi)	ν	σ_Y [1] (10^3 psi)	σ_u [2] (10^3 psi)	α (10^{-6}/°F)
Aluminum	5.2	10	3.8	0.33	3	10	13.3
Aluminum alloys							
2014-T6	5.42	10.6	3.9	0.33	60	68	12.8
6061-T6	5.26	10	3.7	0.33	37	42	13.1
Cast iron alloys							
Gray ASTM 20	14.0	10	3.9	0.28	–	26 (97)	6.7
Malleable ASTM A197	14.1	25	9.8	0.28	–	40 (83)	6.6
Concrete							
Low strength	4.62	3.20	–	0.15	[1.8]	–	6.0
High strength	4.62	4.20	–	0.15	[5.5]	–	6.0
Copper							
Soft (pure)	17.3	16	6.0	0.34	10	32	9.8
Hard (Be-Cu)	16.3	18	6.7	0.34	170	190	9.9
Glass	5.1	10	4	0.25	–	(10–125)	
Magnesium alloy Am 1004-T61	3.54	6.48	2.5	0.30	22	40	14.3
Nickel steel	15	29	11.2	0.29	20–30	45–60	7.2
Plastic–Kevlar 49	2.81	19.0	–	0.34	–	104 (70)	–
Structural steel							
A36	15.2	29	11.3	0.28	36	58	6.7
A572	15.2	29	11.3	0.28	50	65	6.7
Stainless steel							
S40500	15	27.6	10.5	0.31	25	60	9.6
S17700	15	27.6	10.5	0.31	150	185	9.6
Titanium		16	5.8	0.33	60	70	4.7
Titanium alloy 5 Al, 2.5 Sn	8.6	17	6.3	0.34	110	115	5.7
Woods							
Douglas fir	0.97	1.76	–	–	–	(7.4)	–
Southern pine	1.13	1.76	–	–	–	(8.4)	–

1. Shear yield stress noted in brackets [ ].
2. Compression ultimate stress noted in parentheses ( ).

This Page Intentionally Left Blank

Index

A
Absorbers, 475–481
  absorber mass, 476
  damped vibration absorbers, 481
  design criteria, 478
  for control of beam vibration, 712
  forced frequency ratio, 476
  frequency ratio, 477
  mass ratio, 476
  operating range, 477
  optimal tuning of, 482
  primary mass, 476
  undamped vibration absorbers, 476
Acceleration, 280, 284, 307
  angular, 308, 314, 324
  Coriolis, 326
  cylindrical components of, 285
  of mass center, 299, 336
  polar components of, 285
  rectangular Cartesian components of, 285
  relative, 309
  relative to a moving frame, 326
  relative to a translating frame, 324
Action and reaction forces, 280
Active vibration control, 497, 559, 601
Actuator locations, 491, 559
Actuators, 497, 501, 559
Adjoint eigenvalue problem, 452
Admissible displacement, 756
Admissible functions, 582, 585
Air damping, 421
Airy stress function, 761
Amplification factor, 394
Analytical geometry, 864
Analytical methods
  boundary value approach, 19
  convolution integral, 385
  DTFM (*see* Distributed transfer function method)
  Green's function formula, 487
  Laplace transform method, 385
  method of singularity functions, 16
  method of undetermined coefficients, 386
  modal expansion, 537
  stiffness method, 224
Angular acceleration, 308, 314, 324
Angular displacement, 314
Angular momentum, 288, 316, 328
  conservation of, 299,
  of a particle, 288
  of a rigid body, 316, 328
  of a system of particles, 299
Angular velocity, 308, 323
Animation of
  dynamic response of a particle, 290
  dynamic response of a rigid body, 317, 343
  modes of vibration, 446, 534, 627, 680, 814, 821
  transient time response, 554, 642, 701
Approximate methods
  assumed-modes method, 582
  finite element method, 770, 829
  Rayleigh-Ritz method, 582
  Runge-Kutta numerical integration, 290, 423, 465, 501, 567, 701
  truncated modal expansion, 671

Area
beam cross-section, 42
table of, 855
Average normal strain, 145
Average normal stress, 139
Axial force, 56, 204, 620

B
Balancing, of rotating machines, 404
Bars in static equilibrium, 53–83
axial force, 56
bar elements, 67
compatibility conditions, 68
external loads, 62, 71
boundary conditions, 56
distributed transfer function method, 74
governing equation, 56
longitudinal deformation, 56
longitudinal rigidity, 56
MATLAB functions,79
spring constraints, 66
stepped bars, 66
Bars, shafts and strings in vibration, 617–665
animation of modes of vibration, 627, 659
animation of time response, 642
eigenvalue problem, 622
equation of motion, 619
frequency response, 643
forced vibration, 635
free vibration, 632
MATLAB functions, 662
modal expansion, 631
mode shapes, 622
natural frequencies, 622
stepped bars in longitudinal vibration, 648
stepped shafts in torsional vibration, 647
Base excitation, 400
Beam-column, 100
Beam cross-section area, 42
Beam theory, 11
Beams in static equilibrium, 9–51
beam axis, 23
bending moment, 12
bending stiffness, 12
boundary conditions, 13
boundary disturbances, 13, 27
boundary value approach, 19
curvatures of beam axis, 24
distributed transfer function method, 21
equilibrium equations, 12, 25
Euler-Bernoulli, 11
Euler-Bernoulli hypothesis, 23
external loads, 12, 30
flexural rigidity, 24
MATLAB functions, 47
method of singular functions, 16
moment-curvature relation, 24
moment of inertia of area, 42
neutral axis, 23
normal stress and strain, 23
pure bending, 24
reactions at supports, 14
shear force, 14
sign convention, 14
singular functions of, 16
transverse displacement, 11
Beams in vibration, 521–615
animation of modes, 534
animation of time response, 554
boundary conditions, 525, 583
damping, 539
distributed transfer function method, 535
eigenvalue problem, 524
equation of motion, 523
Euler-Bernoulli, 521
external loads, 548
feedback control (*see* Control of beams)
forced vibration, 544
free vibration, 540
frequency response, 555
MATLAB functions, 609
modal coordinates, 538
modal expansion, 537
mode shapes, 524
natural frequencies, 524
nonuniform (*see* Nonuniform beams)
normalized eigenfunctions, 530
rigid-body mode, 526
steady-state vibration, 557
transverse displacement, 523
uniform beams, 523
Beating, 363, 372
frequency of beating, 373
period of beating, 373
Bending moment, 12, 88, 524, 823
Bending stiffness
beams, 12, 524
frame members, 240
plates, 823
Bessel functions, 818, 842
Biharmonic operator, 823, 840
Bi-orthogonality relations, 452
Body forces, 749
Boundary disturbances, 12, 58
Boundary influence functions, 22, 75, 125, 264
Brittle materials, 151
Buckling of columns, 85–133
axial compression load, 87
beam-column, 100
boundary conditions, 89

buckling load, 88
buckling mode shape, 88
characteristic equation, 90
constrained stepped columns, 109
distributed transfer function method, 123
eccentric loading, 96
eigenvalue problem, 88
elastic foundation, 109
Euler buckling load, 88
geometric imperfection, 104
MATLAB functions, 128
nonuniform columns, 117
uniform columns, 87
Bulk modulus of elasticity (*see* Elasticity constants)

C
Calculus of variations, 756
operator of variation, 757
virtual displacement, 756
Center of mass, 299, 310, 327
Central force, 303
Central force motion, 303
conic section, 306
constant areal velocity, 304
Newton's law of gravitation, 304
space vehicle in free flight, 305
trajectories, 306
universal gravitational constant, 304
Characteristic equation
bars, shafts and strings, 623
beams, 526, 673, 718
columns, 90
membranes, 811, 819
multiple-degree-of-freedom systems, 440
one-degree-of-freedom systems, 355
plates, 827, 832, 843
Circular membranes (*see* Membranes)
Circular natural frequency, 353
Circular plates (*also see* Plates), 839
Circular shafts in static equilibrium, 53–83
boundary conditions, 57
compatibility conditions, 68
external loads, 62
governing equation, 57
MATLAB functions, 79
rotation (twist), 57
shaft elements, 68
spring constraints, 69
stepped shafts , 68
torque, 57
torsional rigidity, 57
Circular shafts in vibration (*see* Bars, shafts and strings in vibration)
Closed-loop system, 499, 564
Closed-loop transfer function, 502
Coefficient of friction, 281, 421
Coefficient of linear thermal expansion, 216, 925
Coefficient of restitution, 302
Coefficient of viscous damping, 353, 524
Collision, 302
Colocated control system, 491, 563
Columns (*see* Buckling of columns)
Combined beams in vibration (*see* Constrained, combined and stepped beams in vibration)
Complex eigenvalues, 443, 444
Complex functions, 879
Complex harmonic factor, 473
Complex numbers, 878
Conic sections, 306, 866
Conservation of energy, 288, 316, 335, 380
Conservation of momentum, 299
Conservative force, 287
Conservative system, 287
Constants of Lamé, (*see* Elasticity constants)
Constitutive law (*see* Hooke's law)
Constrained beams in vibration (*see* Constrained, combined and stepped beams in vibration)
Constrained, combined and stepped beams in vibration, 667–736
animation of modes of vibration, 680, 724
animation of time response, 701
attached springs, masses and dampers (SMD), 670
beam segments, 715
boundary conditions, 672
characteristic equation, 673, 718
combined beams, 691–715
constrained beams, 670–691
constraints, 715
eigenvalue locus, 683
eigenvalue problem, 673, 695, 718
elastic foundation, 719
end mass, 717
external loads, 703
frequency response, 687, 707
governing equations, 670, 692
hinged disk, 717
MATLAB functions, 730
modal expansion, 671, 693
mode shape, 673, 696, 716
mounted rigid bodies, 691
natural frequencies, 674, 719
nodes, 715
nonproportionally damped system, 675
oscillators, 691
proportionally damped system, 674
Runge-Kutta method, 701
stepped beams, 715–729
state equation, 700

Constrained, combined and stepped beams in vibration, (*continued*)
transient response, 691, 698
undamped system, 674
vibration absorption, 712
Constrained multispan beams in static equilibrium, 160–200
beam elements, 161
boundary conditions, 163
constraints, 164
distributed transfer function method, 189
external loads, 165, 173
hinge connection, 164
influence lines, 183
MATLAB functions, 195
nodes, 161
reactions at supports, 165
settlement of supports, 178
sign convention, 162
Constraints, 109, 164, 648, 715
Continuous beams (*see* Constrained multispan beams static equilibrium)
Control laws (*see* Feedback control laws)
Control of beams, 559–581
actuators, 560
augmented state equation, 566
closed-loop, 564
discretized model, 561
feedback controllers (*see* Feedback control laws)
open-loop, 562
MATLAB functions, 610
nonuniform beams, 601
pole-zero alternation, 563
poles and zeros, 570
root locus, 574
sensors, 561
state equation, 563
transfer function, 562
Control of mechanical systems (*see* Control of multiple-degree-of-freedom systems)
Control of multiple-degree-of-freedom systems, 487–514
actuators, 491, 497, 501
augmented state equation, 500
closed-loop, 500
feedback controllers (*see* Feedback control laws)
MATLAB functions, 506, 516
open-loop, 491
poles and zeros, 492
position control system, 501
sensors, 491, 497, 501
state representation, 498, 503
transfer function, 491, 500, 502
vibration control system, 497
Control system, 497, 559
Conversion of units, 136, 280
table of, 923
Convolution integral, 385, 881
Coordinate systems, 285
Coordinate transformation, 225, 265, 341
Coordinates
cylindrical, 285
generalized, 582, 594, 605, 671, 687
modal, 538, 631
polar, 285, 765
rectangular Cartesian, 285
Coriolis acceleration, 326
Coulomb friction, 281, 421
Criteria of failure, 151–155
Maximum distortion-energy theory, 152
Maximum normal-stress theory, 153
Maximum shear-stress theory, 151
Critical damping coefficient, 354
Critically damped system, 355
Cross or vector product, 281, 288, 298, 870
Curvature, 24, 824

D
D'Alembert's principle, 315
Damped frequency, 355
Dampers, 252, 438, 670, 692
Damping, 353, 439, 475
air, 421
Coulomb, 281, 421
energy dissipation per cycle, 422
forced response, 361
free response, 358
viscous, 353
Damping coefficient (*see* Coefficient of viscous damping)
Damping form, 440
undamped systems, 355, 442
nonproportionally damped systems, 440, 675
proportionally damped systems, 440, 674
Damping matrix, 439, 674
Damping ratio, 353, 443, 673, 732
Damping status, 355
critically damped system, 355
over-damped system, 355
undamped system, 355
underdamped system, 355
Decoupled differential equations, 450
Deflection
lateral, 55, 87, 810, 823
of support, 205, 243
transverse, 12, 164, 240

Deformation
bending, 23
elastic, 213, 252, 743
longitudinal, 203, 240, 755
of structures, 178
torsional, 55, 755
Degree of freedom, 351, 437
Delta function, 12, 61, 165, 365, 487, 524, 588, 623, 671
Delayed forcing functions, 376, 466
Derivatives, 860
of vectors, 325, 871
table of, 861
Derivatives and integration, 860
Determinant, 874
Diagonal matrix, 874
Dilatation or unit volume change, 750
Dirac delta function (*see* delta function)
Direction cosines, 331, 744
Displacement
angular, 314
boundary, 12, 58
lateral, 621
longitudinal, 56, 203, 240, 620
modal, 124, 454, 533, 589, 658, 723, 815
nodal, 77, 192, 206, 224, 241,266, 771, 831, 918
of elastic bodies, 289, 743
of multiple-degree-of-freedom systems, 439
of one-degree-of-freedom systems, 353
rotational, 68, 308, 649, 693
steady-state, 395, 413, 473, 481, 557, 643, 687, 708
transverse, 11, 88, 161, 240, 523, 671, 810, 823
virtual, 587, 755
Displacement transmissibility, 401
Distortion energy per unit volume, 152
Distributed load or distributed force, 18, 61, 165, 700
Distributed transfer function method, 4, 913
for bars, shafts and strings, 74, 645
for beams, 21, 189, 535
for columns, 123
for frames, 263
for multibody structures, 917
for one-dimensional continua, 913
for two- and three-dimensional continua, 919
Dry friction (*the same as* Coulomb friction)
Dot or scalar product, 287, 870
DTFM (*see* Distributed transfer function method)
Ductile materials, 151
Duhamel integral, 385
Dynamic balance in rotation, 403
Dynamic vibration absorbers (*see* Absorbers)
Dynamic vibration absorption, 475, 712
Dynamics
of bars, shafts and strings, 617
of beams, 521, 667
of particles, 280
of rigid bodies, 307, 323
structural, 519

E
Earth
mass of, 304
radius of, 304
Eigenfunction
bars, shafts and strings, 622, 651
beams, 524, 588, 622, 671, 719
columns, 88
membranes, 811, 818
multiple-degree-of-freedom systems, 450
plates, 827, 845
Eigenfunction expansion (*see* Modal expansion)
Eigenvalue
bars, shafts and strings, 622, 651
beams, 524, 588, 622, 673, 719
columns, 45, 88, 124
matrix, 876
matrix of inertia, 331
membranes, 811, 818
multiple-degree-of-freedom systems, 440, 444
plates, 827, 845
strain matrix, 747
stress matrix, 745
Eigenvalue locus, 683
Eigenvector
bars, shafts and strings
beams, 588, 673, 695
matrix, 876
matrix of inertia, 331
multiple-degree-of-freedom systems, 440, 444
strain matrix, 747
stress matrix, 745
Elastic body, 136, 724, 759
Elastic foundation, 109, 715
Elasticity, 739–805
boundary conditions, 752
boundary tractions, 761, 783
conditions of compatibility, 752
equations of equilibrium, 749
finite element solution, 770
principle of minimum potential energy, 755
problems in polar coordinates, 765
problems of axial symmetry, 767
Saint-Venant's principle, 753
strain-displacement relations, 749
strain energy, 755
stress function, 761

Elasticity, (*continued*)
stress-strain relations, 750
theory of, 742
Elasticity constants, 83, 753
bulk modulus of elasticity, 750
conversion of, 753
Lamé coefficient or constants of Lamé, 750
Poisson's ratio, 149, 750, 808, 823
shear modulus, 55, 149, 620, 750
Young's modulus, 83, 750
Elongation, 142, 281, 744
Energy, 287, 335
conservation of, 288, 316, 335, 380
kinetic, 287, 302, 315, 335, 379, 585, 826, 841
mechanical, 288, 379
potential, 287, 316, 335, 379, 585, 757, 773
strain, 585, 755, 826, 841
Energy dissipation, 303, 422, 475
Engineering materials, 925
Equation of motion
absorbers, 476, 481
axisymmetric rigid bodies, 341
bars, shafts and strings, 619
beams, 523, 587, 670, 692
membranes, 810, 816
multiple-degree-of-freedom systems, 439
one-degree-of-freedom systems, 353, 421
plates, 823, 840
rigid bodies in two dimensions, 314
rigid bodies in three dimensions, 336
single particle, 286, 305
system of particles, 301
Equilibrium equation
bars, shafts and strings, 56
beams, 12, 161
columns, 88, 109, 117
elastic bodies, 749
frames, 240
global, 192, 227, 267, 775
trusses, 204
Equivalent viscous damping coefficient, 422
Euler angles, 340
Euler-Bernoulli beams, 11, 523
Euler buckling load, 88
Euler's equations, 337
Euler's laws, 298
Exponential matrix, 876
bars, shafts, and strings, 76, 646, 652
beams, 22, 190, 537
columns, 123
frame members, 264

F

Failure criteria (*see* Criteria of failure)
Feedback control laws
PD feedback, 503
PID feedback, 499, 504, 565
output feedback, 499, 564
state feedback, 499, 564
Finite element method, 770–801, 829–839
displacement interpolation, 771
element stiffness matrix, 772
for 2-D elasticity problems, 770
for vibration of plates, 829
global stiffness matrix, 775
mesh, 777, 799, 829
nodal displacements, 771, 831
nodes, 770, 829
rectangular elements, 829
shape functions, 771, 831
triangular elements, 771
Flexible beams (*see* Beams in static equilibrium *or* Beams in vibration)
Flexible structures (*see* Structures and structural components)
Flexural rigidity (*the same as* Bending stiffness)
Force
definition of, 281
conservative, 2, 87
external, 297
friction, 281
gravitational, 281
inertia, 315
internal, 297
nonconservative, 287
resultant, 282
spring, 281
Force transmissibility, 395, 401, 404
Forced response or forced vibration
bars, shafts and strings, 635
beams, 544, 701
multiple-degree-of-freedom systems, 454, 469
one-degree-of-freedom systems, 361, 384
Forcing functions, 362, 455
exponential, 369
impulse, 365
ramp, 368
periodic, 410
pulse excitation, 373
sinusoidal, 371
step, 366
Fourier coefficient, 410, 419
Fourier series, 410
of periodic forcing function, 410
of steady-state response, 411
Frame of reference (*see* Reference frame)
Frames, 237–276
coordinate transformation, 265
distributed transfer function method, 263
external loads, 248, 250
global equilibrium equation, 267

internal forces, 241
MATLAB functions, 269
member stiffness matrix, 265
members, 240, 263
nodal displacements, 241
nodal forces, 241
nodes, 240, 267
settlement of supports, 252
static response, 248, 252
supports, 242
Free response or free vibration
bars, shafts and strings, 632
beams, 540, 715
membranes, 809
multiple-degree-of-freedom systems, 453, 459, 469
one-degree-of-freedom systems, 357
plates, 823, 839
Frequency
driving, 477
excitation, 371, 472, 557
of beating, 373
Frequency, natural, (*see* Natural frequency)
Frequency ratio, 394, 477, 483
Frequency response
bars, shafts and strings, 643
beams, 555, 687, 707
multiple-degree-of-freedom systems, 472
one-degree-of-freedom systems, 394, 401, 405
Friction coefficient, 281
kinetic, 281, 421
static, 422
Functional, 757, 826

G
Generalized coordinates, 582, 594, 605, 671, 687
Generalized force, 587
Generalized Hooke's law (*see* Hooke's law)
Global equilibrium equation, 192, 227, 267, 775
Global stiffness matrix, 192, 267, 775
Gradient operator, 287, 871
Gravitation, 304
Newton's law of, 304
universal constant of, 304
Gravitational acceleration, 281
Gravity, 281
Green's function, 22, 75, 124, 190, 263, 487

H
Harmonic excitation
base excitation, 400
external excitation, 400
magnitude, 394, 473
phase angle, 400, 473
resonance, 372
rotating unbalance, 403
steady-state response, 394, 472
vibration absorber, 476, 481
vibration isolators, 476, 481
Hertz (unit), 354, 529, 636
Hooke's law, 149, 750
for plane strain, 150
for plane stress, 150
generalized, 149, 750
Hyperbolic functions, 860

I
Identity matrix, 874
Impact, 302
Impulse, 288, 316, 335
Impulse force, 365, 455, 548, 637, 703
Impulse and momentum, principle of, 288, 315, 335
Impulse response, 385, 487
Inertia
area moments of (*see* Moments of inertia of area)
moments of, 329
polar moment of, 55, 620
principal axes of
principal moments of, 331
products of, 329
properties, 327
Inertia matrix, 439
Inertial force and moment, 315
Influence functions, 451
Influence lines, 183
Initial conditions of
bars, shafts and strings, 631
beams, 524
combined beams, 692, 699
constrained beams, 670
multiple-degree-of-freedom systems, 439
one-degree-of-freedom systems, 361
rigid bodies, 317, 343
single particles, 289
Instantaneous center of rotation, 309
Interactive computing, 2
Integration, 860
table of, 861
Internal force
bars, shafts and strings, 56, 622
beams, 14, 162, 716
columns, 124
frames, 241
plates, 823, 840
systems of particles, 297
trusses, 205, 226
Interpolation, 771, 919
Inverse Laplace transforms, 880, 882
Inverse matrix, 875
Isotropic material, 764

Isolation (*see* Vibration isolation)
Isolator, 406

K
Kinematics, 284, 323
Kinetic energy
  beams, 585
  one-degree-of-freedom systems, 379
  particle, 287
  plates, 826, 841
  rigid bodies, 315, 335
Kinetics
  of particles, 286, 298
  of rigid bodies, 314
Kirchhoff's assumptions, 824
Kronecker delta (*see* delta function)

L
Lagrange's equations, 587
Lamé coefficient (*see* Elasticity constants)
Laplace transforms, 879
  definition of, 879
  for solutions, 385, 458
  inverse Laplace transform, 880,
  table of, 881
Laws of sines and cosines, 859
Laplacian operator, 810, 816
Linear density, 524, 590, 648, 670, 692, 715, 915
Linear independence of vectors, 875
Linear momentum, 298, 316, 327
  conservation of, 299
Logarithms, 854
Longitudinal stiffness or rigidity, 56, 204, 240, 620, 648
Longitudinal vibration of bars (*see* Bars, shafts and strings in vibration)
Lumped parameter systems (*see* Multiple-degree-of-freedom systems)

M
Magnitude of
  force, 281, 405
  frequency response or steady-state response, 394, 401, 422, 473, 481, 557, 644, 687
  vector, 868
  velocity, 287, 308, 335, 395
Mass, 280
Mass center, 299, 310, 327
Mass density, 810, 823, 840
Mass matrix, 439, 694
Mass moments of inertia (*see* Moments of inertia of mass)
Material failure, 151–155
Material properties, table of, 925
Mathematical formulas, 853–888
  algebraic formulas, 853
  analytical geometry, 864
  areas and volumes, 855
  complex numbers, 878
  derivatives and integration, 860
  hyperbolic functions, 860
  Laplace transforms, 879, 882
  matrix theory, 872
  series expansion, 862
  trigonometry, 857
  vector analysis, 868
MATLAB, 5, 889–912
  algebraic and differential equations, 907
  control flow, 903
  Control System Toolbox, 910
  graphics, 898
  mathematical functions, 892
  matrix and vector manipulations, 893
  M-files, 901
MATLAB toolboxes of this book, 3, 5
  bars, shafts and strings in equilibrium, 79
  bars, shafts and strings in vibration, 662
  beams in equilibrium, 47
  beams in vibration, 609
  columns, 128
  constrained, combined and stepped beams in vibration, 730
  constrained multispan beams in static equilibrium, 195
  elastic bodies in equilibrium, 802
  inverse Laplace transforms, 884
  membranes in vibration, 849
  multiple-degree-of-freedom systems, 515
  one-degree-of-freedom systems, 432
  particles and rigid bodies, 349
  plane frames, 269
  plane trusses, 231
  plates in vibration, 849
  stress analysis in two dimensions, 155
Matrix, 872
  determinant of, 874
  eigenvalues and eigenvectors, 876
  exponential of, 877
  identify, 874
  inversion, 875
  rank of, 876
  square, 874
Maximum and minimum normal strains, 144
Maximum distortion-energy theory, 152
Maximum normal-stress theory, 153
Maximum overshoot, 367
Maximum principal strain, 146,
Maximum principal stress, 153, 745
Maximum shear strain, 145
Maximum shear stress, 139, 746
Maximum shear-stress theory, 151

Mechanical energy, 288, 379
Mechanical properties, 925
table of, 925
Membranes, 809–823
boundary conditions, 810, 816
circular, 816
eigenvalue problems, 810, 817
equation of motion, 810, 816
free vibration, 809, 816
MATLAB functions, 849
mode shapes, 812, 818
natural frequencies, 811, 819
nodal circles and nodal diameters, 819
rectangular, 810
Mechanical system, 367, 487, 497, 910
Method of singularity functions, 16
Method of undetermined coefficients, 386
Metric system (*see* Standard International System)
Modal analysis 450, 469, 538, 632
Modal damping, 443, 539, 671, 692
Modal expansion, 450, 491, 537, 560, 631, 671, 693
Modes of vibration, 442, 527, 588, 625, 842
Mode shapes
buckling, 88, 119, 124
vibration, 442, 491, 524, 588, 622, 651, 673, 696, 716, 811, 819, 827, 842
Modulus of elasticity (*see* Elasticity constants)
Mohr's circle for plane strain, 145
Mohr's circle for plane stress, 140
Moment, 281
bending, 12, 88, 524, 823
resultant 282, 314, 336
twisting, 823, 840
Moment equation of motion
in a body frame, 337
in a rotating frame, 336
rotation about a fixed axis, 337
Moment of a force, 281
Moment of inertia of area, 44
Moment of momentum (*the same as* Angular momentum)
Moments of inertia of mass, 310
central axis, 311
matrix of inertia, 330
parallel-axis theorems, 311
principal, 331
principal axes, 331
table of, 312
transformation of, 331
Momentum
angular, 288, 298, 316, 328
conservation of, 299
linear, 288, 298, 316
Motion, 280
central force, 303
of mass center, 301
plane, 308
relative, 309, 324
rotation, 307, 323, 337, 340
rotation about a point, 307
translation, 307
Multiple-degree-of-freedom systems, 437–518
animation of modes of vibration, 446
bi-orthogonality relations, 452
closed-loop, 499, 501, 509
complex eigensolutions, 444
dynamic vibration absorption (*see* Absorbers)
eigenvalue problem, 440
feedback control (*see* Control of multiple-degree-of-freedom systems)
free response, 453, 458, 459
frequency response, 472
forced response, 454, 458, 459
harmonic excitation, 472
influence function, 451
MATLAB functions, 515
modal damping, 443
natural frequencies, 442
nonproportionally damped systems, 440
open-loop, 491
orthonormal relations, 442
proportionally damped systems, 440
Rayleigh damping, 443
rigid-body modes, 439, 452
state eigenvectors, 452
transfer function, 487
undamped systems, 442
Multispan beams (*see* Constrained multispan beams)

N
Natural frequency, 354
a primary system combined with an absorber, 476
bars, shafts and strings, 622, 651
beams, 524, 588, 674, 719
membranes, 811, 819
multiple-degree-of-freedom systems, 442
one-degree-of-freedom systems, 354
plates, 827, 843
Natural period, 354
Neutral axis, 23
Neutral surface, 823
Newton's law of gravitation, 304
Newton's laws of motion, 280
Nodal displacement, 77, 192, 206, 224, 241,266, 771, 831, 918
Nodal line, 340, 819, 843

Nodal points, 491, 533, 563, 589, 628, 723
Nodes, 66, 109, 161, 203, 240, 648, 715, 770, 829, 917
Noncolocated control system, 491, 571
Nonlinear damping
  air damping, 421
  Coulomb damping or dry friction, 421
  equivalent viscous damping coefficient, 422
Nonlinear spring, 421
Nonlinear vibration, 421
Nonproportionally damped system, 440, 675
Nonuniform beams, 581–608
  admissible functions, 582, 585
  boundary conditions, 583
  discretized model, 561
  equation of motion, 581
  kinetic and strain energy, 585
  modes of vibration, 588
Normal strain, 24, 142, 744
Normal stress, 23, 136, 742
Normalized amplitude, 477
Normalized eigenfunctions, 530, 625, 657
Numerical integration, 289, 317, 337, 411, 470, 541, 632, 699, 831
Nutation, 340

O
One-degree-of-freedom systems, 351
  characteristic equation, 355
  characteristic roots, 355
  convolution integral, 385
  critically-damped, 355
  damped frequency, 355
  damping ratio, 353
  forced response, 361
  free response, 357
  frequency response, 394
  harmonic excitation, 394
  Laplace transform, 385
  MATLAB functions, 432
  maximum overshoot, 367
  mechanical energy, 379
  method of undetermined coefficients, 386
  natural frequency, 367
  nonlinear vibration, 421
  overdamped, 355
  response, 365
    to delayed forcing functions, 376
    to exponential force, 369
    to impulse force, 365
    to ramp force, 368
    to periodic excitation, 410
    to pulse excitation, 373
    to sinusoidal excitation, 371
    to step force, 366
  rise time, 367
  Runge-Kutta algorithm, 423
  settling time, 368
  solution by superposition, 377
  undamped, 355
  underdamped, 355
Orthogonal axes, 330
Orthogonal eigenvectors, 444, 675
Orthogonal matrix, 745
Orthonormal relations, 444, 674
Open-loop transfer function, 491, 463
Optimal tuning of absorbers, 482
Oscillators, 691, 712
Overdamped systems, 355

P
Parallel-axis theorems, 311
Partial fraction expansion, 882
Particle dynamics, 280–307
  animation of motion, 290, 317, 343
  dynamic response, 289
  impulse and momentum, 288
  kinematics, 284
  kinetics, 286
  MATLAB functions
  particle system (*see* System of particles)
  work and energy, 287
Particular solution, 21, 386
Passive vibration control, 475
Periodic excitation, 410
Periodic force, 410
  period T of, 410
Phase angle, 394, 473, 893
Plane strain and plane stress, 136, 142, 759
Plates, 823–848
  bending stiffness, 823
  boundary conditions, 825, 841
  circular, 839
  curvatures, 825
  eigenfunctions, 827, 842
  eigenvalue problem, 827
  finite element solution, 829
  free vibration, 823
  kinetic energy, 826
  Kirchhoff's assumptions, 824
  Lévy solutions, 828
  MATLAB functions, 850
  mode shape, 827, 843
  modes of vibration, 842
  moments, 823, 841
  Navier solutions, 827
  natural frequencies, 827, 842
  nodal circles and nodal diameters, 843
  rectangular, 823
  shear forces, 823, 841

strain energy, 826
stress components, 825
Pointwise load or pointwise force, 18, 62, 102, 173, 250, 545, 636, 699, 740
Poisson's ratio (*see* Elasticity constants)
Polar moment of inertia, 55, 620
Poles and zeros, 492, 568, 879
Pole-zero alternation, 492, 563
Position vector, 284, 324, 743
Potential energy, 287, 755
Power, 380, 924
Precession, 340
Pressure, 924
Principal moments of inertia, 331
Principal strains, 144, 746
Principal stresses, 138, 745
Principle of impulse and momentum, 288, 315, 335
Principle of minimum potential energy, 757
Principle of work and energy, 287, 302, 316
Products of inertia, 329
Proportionally damped system, 440, 674
Pulse excitation, 362, 373

Q
Quadratic equations, 854
Quick Solution Guide, 5
buckling of columns, 127
control of beams, 608
control of multiple-degree-of-freedom systems, 515
dynamic analysis of beams, 608,
dynamics of bars, shafts and strings, 661
dynamics of constrained, combined and stepped beams, 729
membranes in free vibration, 802
particle and rigid-body dynamics, 349
plates in free vibration, 802
static analysis of bars, shafts and strings, 78
statics of beams, 46
statics of elastic bodies, 729
statics of multispan beams, 194
statics of plane frames, 268
statics of plane trusses, 230
stress analysis in two dimensions, 155
vibration of multiple-degree-of-freedom systems, 515
vibration of one-degree-of-freedom systems, 431

R
Rayleigh damping, 440
Rayleigh-Ritz method, 582
Reactions at supports, 14, 165, 204, 243, 338
Rectangular membranes (*see* Membranes)
Rectangular plates (*see* Plates)
Reference frame
fixed, 325
moving, 325
rotating, 336
translating, 324
Resonance, 363, 372, 477, 713
Resonant frequency, 712
Resonant vibration, 372, 387, 411, 547, 636, 712
Resultant force and moment, 282
Right-hand rule or screw, 282, 780
Rigid body dynamics, 307, 323
animation of motion, 317, 343
axisymmetric, 340
definition of, 307
Euler angles, 340
Kinematics, 314, 323
linear and angular momentum, 316, 335
MATLAB functions, 349
work and energy, 315, 335
Rigid-body mode, 439, 452, 526
Rise time, 367
Rotation
about a fixed axis, 323
about a point, 307
finite, 324
instantaneous center of, 309
of axisymmetric bodies, 340
of beam cross section, 12
of rigid bodies, 307, 323
Rotating unbalance, 403
Rotational displacement, 308
Runge-Kutta method, 290, 423, 465, 501, 567, 701

S
Saint-Venant's principle, 753
Scalar product (*see* Dot product)
Sensors, 491, 497, 559, 601
Separation of variables, 811, 818, 827
Series expansion
eigenfunction (*see* Modal expansion)
Fourier, 863
Maclaurin, 862
Taylor, 862
Settling time, 368
Shafts in equilibrium (*see* Circular shafts in static equilibrium)
Shafts in vibration (*see* Bars, shafts and strings in vibration)
Shear forces, 14, 88, 162, 241, 524, 582, 671, 823, 840
Shear modulus (*see* Elasticity constants)
SI system (*see* Standard International System)
SI unit prefixes, 923

Sign convention
axial force, 56
bending moment and shear force, 12, 47, 162
external forces, 30, 48, 173
internal forces, 241
reactions at supports, 14
stress components, 136
torque, 57
Signum function, 421, 893
Single-degree-of-freedom systems (*see* One-degree-of-freedom systems)
Singularity function, 17
Solution methods
boundary value approach, 19
distributed transfer function method, 21, 74, 263, 913
finite element method, 770, 829
Laplace transform method, 385, 458
method of singularity functions, 16
Runge-Kutta numerical integration (*see* Runge-Kutta method)
method of separation of variables, 811, 818, 827
principle of superposition (*see* Superposition)
Spin, 340
Spring coefficient, 28, 92, 169, 281, 353, 525, 622, 672, 718
Spring-mass system, 475
Spring-mass-damper system, 353, 441, 491
Stability, 509, 573, 607
Standard International System, 280, 923, 926
State equation, 290, 343, 465, 500, 563, 700, 877
State representation, 498, 503, 572, 605
State space formulation, 500
State variables, 289, 422, 877
State vector, 343, 498, 563, 606
Steady precession, 342
Steady-state response, 376, 394, 410, 422, 476, 557, 916
Step function, 17, 374, 434, 467, 545, 595, 636, 700
Stepped beams in free vibration (*see* Constrained, combined and stepped beams in vibration)
Stiffness
bending, 12, 240, 524, 823
longitudinal, 56, 204, 240, 620, 648
torsional (rigidity), 57, 621
Stiffness matrix, 190, 224, 264, 439, 772, 831
Stiffness method for trusses, 224
Strain, 142, 744
average normal, 145
components, 142, 744
invariants, 747
maximum shear stress, 144, 746
Mohr's circle, 145
normal, 142, 744
principal, 144, 745
shear, 142, 744
transformation, 143
tensor, 745
Strain-displacement relations, 749
Strain energy, 755
density, 755
of elastic members, 756
of nonuniform beams, 585
of plates, 826
Strain gauges, 147
Strain invariants, 747
Stain rosette, 147
Strain transformation, 142, 745
Strength of materials, 7
Stress, 136, 742
average normal, 139
components, 136
invariants, 745
maximum shear, 139
Mohr's circle, 140
normal, 136, 742
principal, 138, 745
shear, 136, 742
tensor, 744
thermal, 764
transformation, 137
units of, 136
Stress analysis, 135, 739
Stress concentration, 769
factor, 769
Stress function, 761
Stress invariants, 745
Stress transformation, 744
Stress-strain relations (*see* Hooke's law)
Strings in static equilibrium, 58–83
external loads, 62
boundary conditions, 58
governing equation, 58
lateral deformation, 58
MATLAB functions, 79
Strings in vibration (*see* Bars, shafts and strings in vibration)
Structural control, 487, 497, 559, 601
Structural dynamics, 519
Structural mechanics, 157
Structures and structural components
bars, shafts and strings, 53, 617
beams, 9, 521
columns, 85
constrained and combined beams, 667
elastic bodies, 135, 739
frames, 237

membranes, 809
multispan beams, 159
plates, 823, 839
trusses, 201
Superposition, 65, 101, 256, 377, 416, 468, 640
System of particles, 297–303
collision, 302
Euler's laws, 298
impulse and momentum, 298
mass center, 299, 301
work and energy, 302

T
Thermal stresses, 764
Thick cylinder, 768
Time derivatives of vectors in a moving reference, 325
Time response
bars, shafts and strings, 630
beams, 537
constrained and combined beams, 698
multiple-degree-of-freedom systems, 450
one-degree-of-freedom systems, 357
particles, 289
rigid bodies, 317, 343
Torque-free motion, 342
Torsion of circular shafts (*see* Circular shafts in static equilibrium)
Torsional rigidity, 57, 620,
Torsional vibration of shafts (*see* Bars, shafts and strings in vibration)
Transfer function, 385, 487, 500, 563, 602
Transfer function formulation, 487, 601
Transformation equation
for matrix of inertia, 331
for strain, 137, 745
for stress, 143, 744
for stiffness matrix, 226, 266
Transient response (*see* Time response)
Transmissibility
displacement, 401
equation for isolator design, 408
force, 395, 401, 404
Transmitted force, 395, 404
Transmitted nodal force, 191, 265, 918
Transverse displacement, 11, 523, 823
Trigonometric functions, 857
Trigonometric identities, 858
Trusses, 201–235
axial (internal) force, 204
coordinate transformation, 225
fabrication errors, 216
external forces, 210
MATLAB functions, 231
member, 203
member stiffness matrix, 224
nodal displacements, 206
nodes, 203
static response, 210
stiffness method, 224
support settlement, 213
supports, 204
thermal effects, 216
total response, 220

U
Ultimate normal stress, 153
Undamped system, 355, 442
Underdamped system, 355
Unit conversion (*see* Conversion of units)
Unit volume change or dilatation, 750
U.S. customary units, 136, 280, 923

V
Variational calculus (*see* Calculus of variations)
Vectors, 868
addition and subtraction of, 869
components of, 869
cross or vector product, 870
curl, 872
divergence, 871
dot or scalar product, 870
gradient, 872
magnitude
scalar triple product of, 971
Velocity, 284
absolute, 328
angular, 308, 323, 328
cylindrical components of
instantaneous center of
polar components of
rectangular components of
relative, 328
relative to a fixed frame, 325
relative to a moving frame, 325
Vibration, 351, 437
absorption, 475
analysis, 384
control, 497, 559
damped, 355, 443
forced, 361, 544
free, 357, 540
isolation, 406
nonlinear, 421
undamped, 358, 366, 450
Vibration absorbers (*see* Absorbers)
Vibration absorption, 475, 712
Vibration isolation, 406
base excitation, 407
isolators, 406

Vibration isolation, (*continued*)
rotating unbalance, 407
transmissibility formulas, 407
Vibration isolators (*see* Vibration isolation)
Vibration of multiple-degree-of-freedom systems (*see* Multiple-degree-of-freedom systems)
Vibration of single-degree-of-freedom systems (*see* One-degree-of-freedom systems)
Virtual displacement, 587, 756,
Virtual work, 586, 757
Viscous damping, 353, 421, 481, 524, 539, 671, 692
Volume of three-dimensional shapes, 856

W
Weight, 281, 305
Windows, 5
Work, 287
Work and energy, principle of, 287, 302, 316

Y
Yield stress, 152
Young's modulus (*see* Elasticity constants)

Z
Zeros of a transfer function, 492, 563, 603